uc3m | Universidad Carlos III de Madrid

Trabajo Fin de Grado

Desarrollo de aplicación “Serious Game” para rehabilitación mano

Grado en Ingeniería Electrónica Industrial y Automática

2016-2017

Autor

Ángel Fernández Barroso

Tutor

Edwin Daniel Oña Simbaña

Madrid 26 - septiembre - 2017

RESUMEN

La realización de este proyecto surge de la necesidad de encontrar una forma económica y fiable de realizar la rehabilitación del túnel carpiano, en pacientes que presentaban dicho síndrome, y, que han sido sometidos a un procedimiento quirúrgico en el que se acorta el ligamento que ejerce presión sobre dicho nervio.

En los métodos de rehabilitación convencionales se requiere de una persona cualificada que evalúe, indique y ayude al paciente en todo momento a hacer ciertos movimientos que sirven para conseguir que el sujeto pueda llevar una vida normal, permitiéndole realizar todas las acciones cotidianas en las cuales se ve implicado el túnel carpiano.

Con la premisa de reducir costes, se ha automatizado parte del trabajo de rehabilitación, ayudando así al trabajo del rehabilitador. A su vez se consigue, con este trabajo, una mayor flexibilidad, puesto que se consigue que el paciente pueda realizarlo en casa simplemente teniendo un ordenador, el *Leap Motion*, que, gracias a su software rastrear cada mano con sus respectivos huesos y darnos información de los mismos, y el programa creado. Gracias a ello se ahorra en el espacio necesario para la rehabilitación y en el tiempo de dedicación a la misma por parte del fisioterapeuta, a la vez que se aumenta la comodidad del paciente pues se evitan traslados al centro y puede realizarse en el momento más oportuno para él.

Para ello se ha diseñado un programa con una serie de sistemas de evaluación con los que poder cuantificar el movimiento que puede realizar el paciente, tanto de la muñeca como el movimiento de los distintos dedos y sus respectivas falanges. Estos datos pueden ser utilizados para medir el avance del paciente en la rehabilitación. A su vez, y a partir de los datos obtenidos en la evaluación, se han diseñado varios videojuegos que obligan al paciente a mover la parte afectada para conseguir así un aumento gradual en los movimientos que es capaz de realizar tras la operación de una manera más amena.

Palabras clave: *Leap Motion*, *serious game* y rehabilitación

ABSTRACT

The accomplishment of this project arises from the need to find an economic and trustworthy way of realizing the rehabilitation of the tunnel carpiano, in patients who were presenting the above mentioned syndrome, and, that have been submitted to a surgical procedure in the one that is slow the ligament that exercises pressure on the above mentioned nerve.

In the conventional methods of rehabilitation it is needed of a qualified person who evaluates, indicate and help the patient to do at all time certain movements that serve to achieve that the subject could take a normal life, allowing him to realize all the daily actions in which the tunnel meets implied carpiano.

With the premise of reducing costs, there has been automated part of the work of rehabilitation, helping this way to the work of the rehabilitator. In turn a major flexibility is obtained, with this work, since there is achieved that the patient could realize it in he marries simply having a computer, the Leap Motion, that, thanks to his software to trace every hand with his respective bones and to give us information of the same ones, and the created program. Thanks to it it saves itself in the space necessary for the rehabilitation and in the time of dedication to the same one on the part of the physical therapist, simultaneously that increases the comfort of the patient since movements are avoided to the center and can be realized in the most opportune moment for him.

For it a program has been designed by a series of systems of evaluation with which to be able to quantify the movement that the patient can realize, so much of the wrist like the movement of the different fingers and his respective phalanxes. This information can be used to measure the advance of the patient in the rehabilitation. In turn, and from the information obtained in the evaluation, there have been designed several video games that force the patient to move the part affected to obtain this way a gradual increase in the movements that it is capable of realizing after the most pleasant operation of a way.

Key words: Leap Motion, serious game and rehabilitation

ÍNDICE

RESUMEN 3
ABSTRACT 4
ÍNDICE 5
ÍNDICE DE FIGURAS 9
ÍNDICE DE TABLAS 11
ÍNDICE DE FLUJOGRAMAS 11
ÍNDICE DE IMÁGENES 12
ÍNDICE DE GRÁFICOS 12
1. INTRODUCCIÓN 14
1.1. MOTIVACIÓN 15
1.2. OBJETIVOS DEL PROYECTO 16
1.3. DESARROLLO CRONOLÓGICO 17
1.4. ESTRUCTURA Y ORGANIZACIÓN DE LA MEMORIA 18
1.5. HERRAMIENTAS UTILIZADAS 19
1.5.1. HARDWARE 19
1.5.2. SOFTWARE 27
2. ESTADO DEL ARTE 29
2.1. REHABILITACIÓN TRADICIONAL 29
2.1.1. REHABILITACIÓN DEL SÍNDROME DEL TUNEL CARPIANO 30
2.2. SERIOUS GAMES 32
2.2.1. PULSE!! 33
2.2.2. SIMULADOR FORMULA 1 34
2.3. SISTEMAS DE CAPTURA DE MOVIMIENTOS 35
2.4. UTILIZACIÓN DEL LEAP MOTION EN LA MEDICINA 37

3. DESARROLLO DEL TFG 40
3.1. VIDEOJUEGOS INICIALES 40
3.1.1. ROLL A BALL 40
3.1.2. PLATAFORMAS 2D 44
3.1.3. SPACE SHOOTER 46
3.2. SISTEMA DE EVALUACIÓN 48
3.2.1. EVALUACIÓN MANOS 49
3.2.2. EVALUACIÓN DEDOS 55
3.3. SERIOUS GAMES 57
3.3.1. SPACE SHOOTER 2.0 57
3.3.2. JUEGO GRANJA 62
3.4. INTERFAZ 68
3.4.1. EXPLICACIONES 70
3.4.2. MENU_INICIAL 71
3.4.3. GUARDADO 78
4. RESULTADOS 80
4.1. CÁLCULO ÁNGULOS 80
4.1.1. ÁNGULOS MANO 80
4.1.2. ÁNGULOS DEDOS 85
4.2. PRECISIÓN ÁNGULOS MANO 88
4.2.1. DISTANCIA MANO 88
4.2.2. SOPORTE 94
4.3. PRECISIÓN ÁNGULOS DEDOS 98
4.3.1. ÁNGULO MÁXIMO PROXIMALES 98
4.3.2. POSICIÓN DIGITAL VS POSICIÓN REAL 101
4.3.3. ÁNGULO MÁXIMO MEDIOS 103
4.3.4. RESULTADOS DEDOS 104

4.4. RESULTADOS EVALUACIONES 105
4.5. RESULTADOS JUEGOS 105
5. CONCLUSIONES Y TRABAJOS FUTUROS 106
5.1. CONCLUSIONES 106
5.2. FUTUROS TRABAJOS 109
6. PRESUPUESTO Y MARCO REGULADOR 110
6.1. PRESUPUESTO 110
6.1.1. COSTE PERSONAL 110
6.1.2. COSTE MATERIAL 111
6.1.3. COSTE TOTAL 112
6.2. IMPACTO SOCIO-ECONÓMICO 113
6.3. MARCO REGULADOR 114
7. BIBLIOGRAFÍA 115
8. ANEXO 119
8.1. MENU_INICIAL 119
8.1.1. INICIO SCRIPT 119
8.1.2. CREAR JUGADOR SCRIPT 122
8.1.3. CARGAR JUGADOR SCRIPT 123
8.2. PRUEBA 1 125
8.2.1. BOTONES 125
8.2.2. EVALUACIÓN 125
8.3. PRUEBA 2 131
8.3.1. BOTONES 2 131
8.3.2. EVALUACION 2 132
8.4. MAIN 137
8.4.1. BOTONES NAVES 137
8.4.2. DESTROY BY BOUNDARY 137

8.4.3. DESTROY BY CONTACT 137
8.4.4. GAME CONTROLLER 138
8.4.5. MOVER 145
8.4.6. MOVER ASTEROIDE 145
8.4.7. PLAYER CONTROLLER 145
8.4.8. RANDOM ROTATION 146
8.5. TOPOS 147
8.5.1. BOTONES SLIDERS 147
8.5.2. DESTROY BY BOUNDARY 2 147
8.5.3. DESTROY BY CONTACT 147
8.5.4. GAME CONTROLLER 2 148

ÍNDICE DE FIGURAS

Figura 1. 1. Leap Motion 20
Figura 1. 2. Partes del Leap Motion 21
Figura 1. 3. Ángulo de cobertura del Leap Motion 21
Figura 1. 4. Aplicación principal del Leap Motion 23
Figura 1. 5. Leap Motion en Oculus Rift 24
Figura 1. 6. Soporte Leap Motion 25
Figura 1. 7. Ordenador ASUS X550VX 26
Figura 1. 8. Logo Unity 27
Figura 1. 9. Logo Visual Studio 28
Figura 2. 1. Pinzamiento nervio mediano 31
Figura 2. 2. Operación túnel carpiano 31
Figura 2. 3. Juego Pulse!! 33
Figura 2. 4. Simulador de F1 34
Figura 2. 5. Cuerpo humano digitalizado 36
Figura 2. 6. Traje MoCap 37
Figura 2. 7. MotionSavy´s 38
Figura 2. 8. Utilización software Vivid Visión 38
Figura 3. 1. Roll a Ball Nivel 1 43
Figura 3. 2. Roll a Ball Nivel 2 43
Figura 3. 3. Roll a Ball Nivel 3 44
Figura 3. 4. Personaje Plataformas 2D 45
Figura 3. 5. Escenario Plataformas 2D 45
Figura 3. 6. Naves y Asteroides 46
Figura 3. 7. Flexión y Extensión 49
Figura 3. 8. Abducción y Aducción 50

Figura 3. 9. Movimiento vertical 51
Figura 3. 10. Texto Final Mano 51
Figura 3. 11. Ángulos Dedos 55
Figura 3. 12. Evaluación Dedos 2 56
Figura 3. 13. Game Over Space Shooter 2.0 58
Figura 3. 14. Modificaciones dificultad 62
Figura 3. 15. Manzana Juego Granja 64
Figura 3. 16. Cerdo Juego Granja 64
Figura 3. 17. Escenas del Serious Game 68
Figura 3. 18. Mano con Barra 69
Figura 3. 19. Juego Granja con explicación 71
Figura 3. 20. Canvas del Menu_Inicial 71
Figura 3. 21. Menu_Inicial Inicio 72
Figura 3. 22. Menu_Inicial Jugador 73
Figura 3. 23. Menu_Inicial CrearJugador 74
Figura 3. 24. Menu_Inicial CargarJugador 74
Figura 3. 25. Menu_Inicial Juegos 75
Figura 3. 26. Archivo Excel 78
Figura 3. 27. Datos paciente 78
Figura 3. 28. Ángulos Mano 79
Figura 3. 29. Ángulos Dedos 79
Figura 3. 30. Tiempo Topos 79
Figura 4. 1. Vectores Hand 81
Figura 4. 2. Lógica cálculo ángulos manos 81
Figura 4. 3. Coordenadas Unity 82
Figura 4. 4. Coordenadas Unity con las manos 82
Figura 4. 5. Coordenadas calculo ángulo manos 83

Figura 4. 6. palmVector 83
Figura 4. 7. Coordenadas Giro derecha e izquierda 84
Figura 4. 8. Código ángulo mano 85
Figura 4. 9. Huesos Mano 86
Figura 4. 10. Vector dedos Unity 87
Figura 4. 11. Coordenadas Dedos Mano 88
Figura 4. 12. Posición 90º Mano Digital 101
Figura 4. 13. Posición cercana a 90º Mano Digital 102
Figura 4. 14. Posición 90º falange media real 103

ÍNDICE DE TABLAS

Tabla 1. 1. Características Ordenador 27
Tabla 6. 1. Horas por Fase 110
Tabla 6. 2. Coste de personal 111
Tabla 6. 3. Coste imputable Hardware 112
Tabla 6. 4. Coste Proyecto 112

ÍNDICE DE FLUJOGRAMAS

Flujograma 3. 1. Juego Roll a Ball 42
Flujograma 3. 2. Movimiento Nave 47
Flujograma 3. 3. Movimiento Bala 48
Flujograma 3. 4. Cálculo ángulo Flexión Extensión 52
Flujograma 3. 5. Calculo ángulo Abducción y Aducción 54
Flujograma 3. 6. Movimiento Nave con la mano 59

Flujograma 3. 7. Lógica Space Shooter Inicial 60
Flujograma 3. 8. Lógica Disparar 60
Flujograma 3. 9. Nave colisión asteroide 61
Flujograma 3. 10. Fin Juego Naves 61
Flujograma 3. 11. Juego Cazar Topo 66
Flujograma 3. 12. Fallos Granja 67
Flujograma 3. 13. Menu_Inicial 76
Flujograma 3. 14. Representación Formas 77
Flujograma 3. 15. Botón Exit 77

ÍNDICE DE IMÁGENES

Imagen 3. 1. Posición mano falanges proximales 63
Imagen 3. 2. Posición mano falanges medias 63
Imagen 4. 1. Posición 90º Mano Real 101
Imagen 4. 2. Posición 90º falange media real 102

ÍNDICE DE GRÁFICOS

Gráfico 4. 1. Subida y Bajada a menos de 10 cm 89
Gráfico 4. 2. Subida y Bajada a 20 cm 90
Gráfico 4. 3. Subida y Bajada a 25 cm 90
Gráfico 4. 4. Subida y Bajada a 30 cm 91
Gráfico 4. 5. Derecha e Izquierda a menos de 10 cm 92
Gráfico 4. 6. Derecha e Izquierda a 15 cm 92

Gráfico 4. 7. Derecha e Izquierda a 20 cm 93
Gráfico 4. 8. Derecha e Izquierda a 25 cm 93
Gráfico 4. 9. Derecha e Izquierda a 30 cm 94
Gráfico 4. 10. Subir y Bajar a 25 cm con apoyo 95
Gráfico 4. 11. Subir y Bajar a 25 cm sin apoyo..... 95
Gráfico 4. 12. Ángulo Máximo con apoyo extensión..... 96
Gráfico 4. 13. Ángulo Máximo sin apoyo extensión..... 97
Gráfico 4. 14. Ángulo Máximo Proximales 99
Gráfico 4. 15. Mano a más de 90º 99
Gráfico 4. 16. Ángulo Máximo Medios 103

1. INTRODUCCIÓN

La sociedad actual en la que nos encontramos es conocida como Era de la Información o Era Digital. Esto es debido a que este período de la historia va ligado a las tecnologías de la información y la comunicación. En esta sociedad, el mundo de los videojuegos tiene un papel muy importante gracias al crecimiento exponencial que está experimentando y por el cual se está consiguiendo un gran realismo gráfico y una cada vez mayor interacción entre el jugador y el juego.

Este sector en auge no solo es utilizado en la actualidad con fines lúdicos, sino que gracias al avance que ha tenido hace que los videojuegos tengan cada vez más finalidades. Esto se puede observar en la creación de los llamados "juegos serios" ("serious games" en inglés). Estos juegos, cuya finalidad es la adquisición de conocimientos y práctica, son utilizados por un gran número de personas. Ciertos sectores profesionales, que en su trabajo necesitan tener una gran precisión y una excelente capacidad óculo-manual, utilizan videojuegos para practicar y mejorar su capacidad, como pueden ser los simuladores de vuelo que utilizan los pilotos o los simuladores de operaciones que son utilizados por médicos para realizar prácticas de forma segura.

A su vez hay una tendencia al alza en el desarrollo y uso de aparatos que permiten detectar movimientos realizados por el cuerpo humano, consiguiendo así una interacción cada vez mayor y más realista entre el videojuego y el jugador.

De la unión de los serious games con estos controladores surge la rehabilitación con videojuegos. Este tipo de terapia, que hace uso de la realidad virtual para su desarrollo, permite realizar ejercicios con un componente más lúdico y motivador que los tradicionales. Además, con ello se consiguen captar el movimiento y posición de las partes del cuerpo que se requiera, pudiendo parametrizar estos movimientos obteniendo así una información cuantitativa que la persona que está controlando la terapia no es capaz de captar.

En este Trabajo Fin de Grado (TFG) se ha programado un videojuego que será de utilidad en el proceso de rehabilitación postoperatorio del síndrome del túnel carpiano. Esto se ha conseguido haciendo uso del *Leap Motion* y de un software de código abierto.

1.1. MOTIVACIÓN

La realización de este proyecto está dentro del marco de actividades del proyecto *RoboHealth*. Este equipo tiene como objetivo reducir el cada vez más elevado número de personas necesarias para la rehabilitación. Esto se pretende conseguir con el uso de robots de asistencia y rehabilitación de personas con movilidad limitada, habilidades cognitivas reducidas o enfermedades crónicas y el fin que persigue es que dichas personas obtengan una mayor calidad de vida aportándoles toda la independencia posible [1].

Este *serious game* surge como solución a la posibilidad de mejorar la métrica que se puede obtener de las manos y el sistema de rehabilitación del síndrome del túnel carpiano. Esta enfermedad actualmente afecta al 10 % de la población mundial, especialmente a aquellos activos laboralmente [2]. El aumento progresivo de dicha dolencia genera un aumento en la cantidad de profesionales necesarios para la rehabilitación y, gracias a este trabajo, se pretende conseguir una menor implicación en tiempo y una mayor precisión en la evaluación por parte del terapeuta.

Con este estudio se quiere comprobar, en primer lugar, la precisión de los datos obtenidos por el *Leap Motion*. Esto es necesario para poder estudiar la fiabilidad del aparato y poder así hacer un juicio óptimo de los resultados obtenidos. Una vez realizado eso se pretende conseguir que el paciente que realiza la rehabilitación esté lo más cómodo posible y, en la medida de lo posible, no piense que es una terapia obligatoria, si no que sea algo entretenido con lo que obtener un *feedback* positivo del paciente. Otro punto que pretende conseguir este proyecto es dar al terapeuta cierta información, que el ojo humano no puede captar, para que la evalúe y pueda llevar un mayor control de la evolución del paciente y de sus movimientos.

En este trabajo se ha escogido la última tecnología disponible en el mercado, teniendo en cuenta la premisa de contar con un precio bajo y software de código abierto para conseguir así crear un videojuego asequible y accesible que pueda ser utilizado en cualquier lugar, no solo en el centro médico. Esto a su vez permite a los pacientes realizar los ejercicios en casa, abaratando el trabajo de rehabilitación.

1.2. OBJETIVOS DEL PROYECTO

El objetivo final del TFG "Desarrollo de aplicación *serious game* para rehabilitación de mano" es la creación de una aplicación que permita complementar o sustituir a la actual rehabilitación que se realiza en los pacientes que han sido operados por padecer el síndrome del túnel carpiano. Para poder llevar a cabo este objetivo se han tenido en cuenta los siguientes objetivos secundarios:

- Comprobar los datos que podemos obtener del *Leap Motion*. Esto es necesario para poder realizar un videojuego acorde a lo que podemos medir y a su vez para saber qué información puede ser dada al terapeuta

- Analizar la fiabilidad y precisión del *Leap Motion*. Con este objetivo se pretende evaluar este aparato, del cual obtenemos información acerca de la posición de la mano, para conocer la fiabilidad de los datos obtenidos y actuar en consecuencia a ello, pudiendo despreciar ciertos movimientos por la imprecisión en los mismos.

- Crear un sistema de rehabilitación para todas las edades.

- Conseguir un entorno dinámico y motivador que incentive el uso de juegos y por tanto mejore el proceso de rehabilitación.

- Conseguir un sistema de rehabilitación más económico que la rehabilitación tradicional al no ser necesario que haya un terapeuta para su realización.

- Modernizar el sistema de rehabilitación actual almacenándose de forma automática los resultados obtenidos en un archivo al que tenga acceso el terapeuta.

- Poder valorar la condición física del paciente mediante la cuantificación de los rangos de movimiento que puede hacer.

- Por último, y tal vez más importante, el objetivo que se busca con este proyecto es llevar la rehabilitación a las casas de los pacientes. Esto solo se puede conseguir con un dispositivo económico con gran precisión como es el *Leap Motion*, lo que

permite que se pueda realizar el trabajo del paciente más frecuentemente por eliminar los desplazamientos y esperas que se tendrían al tener que ir a la clínica de rehabilitación cada vez. También es necesario una interfaz fácil de usar y de entender por todos los públicos.

1.3. DESARROLLO CRONOLÓGICO

Para poder llevar a cabo los objetivos explicados en el apartado anterior se ha seguido un orden, el cual será expuesto de forma cronológica para poder entender como se ha llegado al objetivo final.

1. Análisis y estudio de las características del dispositivo *Leap Motion*.

2. Elección del motor de videojuegos en función de la interoperabilidad con el *Leap Motion*.

3. Aprendizaje del lenguaje de programación necesario para programar en Unity (En este caso C#).

4. Programación de videojuegos con el motor de videojuegos Unity para familiarización del mismo.

5. Utilización del *Leap Motion* dentro del entorno gráfico de Unity.

6. Desarrollo de videojuegos en Unity utilizando el dispositivo *Leap Motion* en ellos.

7. Desarrollo de un sistema de evaluación para comprobar las métricas que se pueden obtener de las manos gracias al dispositivo.

8. Comprobación de los datos obtenidos teniendo en cuenta su fiabilidad y precisión.

9. Estudio de la necesidad de utilizar un soporte para un mayor control y ergonomía del elemento.

10. Desarrollo de los distintos sistemas de evaluación con su correspondiente guardado de métricas obtenidas.

11. Creación de una interfaz para acceder a los distintos pacientes y a las opciones que contiene el *serious game*.

12. Desarrollo de los videojuegos que utilicen el *Leap Motion* como sistema de rehabilitación.

13. Modificación de los videojuegos para facilitar su uso por pacientes.

14. Redacción de la memoria.

1.4. ESTRUCTURA Y ORGANIZACIÓN DE LA MEMORIA

La estructura que sigue la memoria está organizada siguiendo ciertas pautas que van a ser explicadas a continuación, incluyendo una breve descripción de los distintos apartados de los que consta la misma.

En primer lugar, se nombran los elementos que son utilizados para la realización de este trabajo, justificando en cada caso su utilización. Para ello se dividirá en dos apartados, siendo el primero de ellos el hardware utilizado y un segundo el software. En ellos se explica la elección de cada uno de los elementos en función de las necesidades y características de los mismos.

Tras esto, se explica el estado del arte, en el cual se referencian los distintos temas que son abarcados en este trabajo. Aquí se describirá la rehabilitación tradicional, el cambio que supone el hecho de utilizar una rehabilitación con videojuegos y el elemento principal utilizado durante el desarrollo de esta nueva terapia de rehabilitación, que es el

Leap Motion, explicando cómo está siendo utilizado este y otros sistemas para automatizar la medicina.

El siguiente punto que exponer es el proyecto realizado. En este punto se explicarán las métricas obtenidas y la fiabilidad de las mismas, para posteriormente describir el *serious game* desarrollado en función de las métricas obtenidas, así como los trabajos previos realizados y la programación necesaria para conseguir todo ello.

Una vez conocido el proyecto se realiza un estudio económico y de las posibles ventajas y cambios que puede provocar el mismo en la sociedad actual y en los métodos utilizados en este momento. Posteriormente se explica el marco regulador en el que se engloban para explicar la información necesaria sobre la propiedad intelectual de la idea.

Por último, se expondrán las conclusiones extraídas gracias a la realización del proyecto, amén de posibles trabajos futuros que se puedan realizar tras este.

1.5. HERRAMIENTAS UTILIZADAS

Dentro del trabajo se ha tenido que hacer uso de una serie de herramientas para el correcto funcionamiento del mismo. Estos elementos se pueden diferenciar en Hardware y Software.

1.5.1. HARDWARE

El Hardware hace referencia a la parte física de cualquier ordenador o sistema informático [3]. En el caso que nos atañe podemos diferenciar los siguientes elementos: *Leap Motion*, soporte, y ordenador.

1.5.1.1. Leap Motion

El *Leap Motion* es un dispositivo de reducidas dimensiones (75mm de largo, 25 mm de ancho y 11 mm de alto) que permite reconocer la posición y los movimientos de

nuestras manos gracias a una imagen virtual que crea en el ordenador [4]. La conexión con el mismo se realiza por medio de un puerto USB. Esta herramienta salió al mercado en el año 2013 a través de la empresa con el mismo nombre, LEAP MOTION.

Figura 1. 1. Leap Motion

La composición del mismo es de dos cámaras, tres leds y un microcontrolador.

1. Cámaras: estas son una de las partes más importantes del dispositivo, pues son las encargadas de la captura de imágenes y por tanto el funcionamiento de las mismas condiciona al resto del sistema. Cada una de las dos cámaras consta de un sensor monocromático, sensible a la luz infrarroja, con una longitud de onda de 850 nm. Dicho sensor es de tipo CMOS para una correcta digitalización de los píxeles y un bajo consumo.

2. Leds: estos son los encargados de la iluminación infrarroja. La longitud de onda a la que trabajan es la misma que la de las cámaras (850 nm). La iluminación que producen depende de la luz que haya en el ambiente pudiendo variar su consumo eléctrico para regularlo.

3. Microcontrolador: se trata de un circuito integrado utilizado para hacer la función de BIOS. Es el encargado de regular la iluminación y de recoger la información para enviarla al driver.

Figura 1. 2. Partes del Leap Motion [4]

La zona de cobertura que tiene este dispositivo es el de una semiesfera de 61 cm de radio, teniendo un ángulo de visión horizontal y vertical de 150, 92º. Lo que implica que la zona de interacción no llegue a ser una semiesfera, pues esta sería de 180º. Véase figura 1.3. Pudiendo rastrearse las manos u otros elementos que se encuentren dentro de la zona de cobertura.

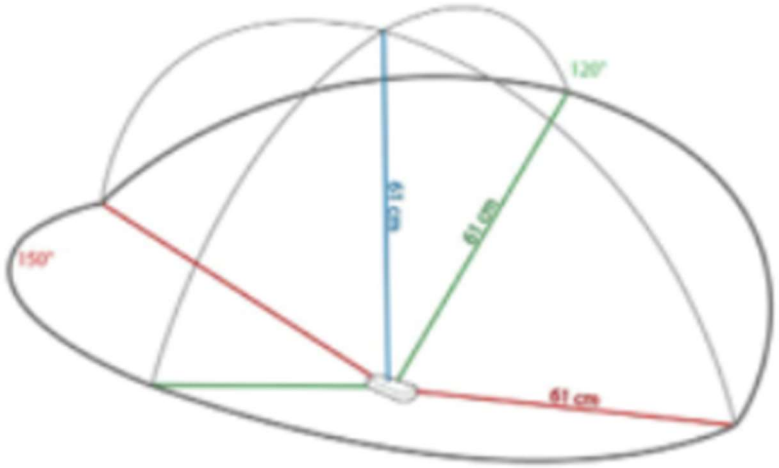


Figura 1. 3. Ángulo de cobertura del Leap Motion [4]

Este dispositivo nos permite obtener información tridimensional, lo cual es necesario para realizar un correcto rastreo de la posición de las manos. Esto nos sirve para identificar correctamente los gestos que se hacen.

En un principio, el *Leap Motion* no estaba pensado como un elemento de rehabilitación puesto que su funcionalidad consistía en permitir la interacción mediante gestos manuales de la persona con el ordenador. Pero a raíz de su salida a mercado se han encontrado multitud de usos, siendo la traducción del lenguaje de signos una de las más relevantes.

La elección de este instrumento se basa en las características que este tiene. Dicho objeto tiene un precio reducido, aproximadamente de 60 €, lo cual hace de él uno de los dispositivos de detección de rastreo de manos más económico que existe. Además, la precisión del mismo no se ve afectada por ello. Según un estudio realizado en 2013 la sensibilidad que tiene permite detectar movimientos de hasta 0,7mm.

De esta herramienta se puede sacar grandes cantidades de información respectiva a las manos como es la posición y velocidad de la palma de la mano o la dirección que tiene cada falange en todo momento, así como las dimensiones de los distintos huesos que componen la mano. Este tipo de información puede llegar a resultar muy útil puesto que no es posible su rastreo por parte del ojo humano.
Se pueden detectar objetos que porten los jugadores en las manos con gran precisión. Por otra parte, una de las ventajas que tiene es el hecho de que no es necesario portar ningún objeto para su uso, haciendo que pueda ser utilizado por personas de cualquier rango de edad y sin necesidad de tener un amplio rango de movilidad siempre y cuando puedan mover la mano.

Otro de los aspectos para tener en cuenta es su metodología Plug and Play [5]. Gracias a ello con enchufar el dispositivo al ordenador y descargarnos de la página oficial del Leap la aplicación podemos empezar a utilizarlo como se ve en la figura 1.4. Los drivers vienen ya en el dispositivo.

Figura 1. 4. Aplicación principal del Leap Motion[4]

Lo siguiente a destacar es la cantidad de aplicaciones descargables que tiene este dispositivo, teniendo su propio *App Market* del que se pueden descargar aplicaciones con distintas finalidades, siendo algunas de pago y otras gratuitas [6]. Esto es debido a la gran capacidad de programación y de desarrollo que permite el dispositivo que hace que sea posible que muchos programadores los utilicen para crear sus propias aplicaciones. El reconocimiento de gestos que permite el SDK del *Leap Motion* es bastante amplio y en la actualidad está disponible tanto para Windows como Linux y OSX. Por otra parte, los lenguajes de programación que son admitidos son C++, C#, Java, JavaScript y Python. En la página oficial del *Leap Motion* se puede encontrar la documentación necesaria para cada uno de ellos. Así como información de las plataformas a las que se les puede integrar el controlador para utilizar en realidad virtual, que son Unity y Unreal. Para poder trabajar con ellas es necesario descargarse un software gratuito que se encuentra en la misma página que es el *Orion Beta*. Existen otros paquetes de *Assets* que nos dan la oportunidad de incluir funciones extra a nuestro dispositivo como son el detectar colisiones en realidad virtual.

Por último, cabe destacar, que se puede usar con las gafas de realidad virtual *Oculus Rift* de realidad virtual, las cuales aunque aumentan el precio del dispositivo,

pueden dotar de mayor realismo y dinamismo a las aplicaciones creadas como se puede ver en la figura 1.5.

Figura 1. 5. Leap Motion en Oculus Rift

1.5.1.2. Soporte

Uno de los principales problemas que hay a la hora de recoger medidas de la posición de la mano y de los ángulos que esta tiene es que cada vez que se recojan dichas medidas no tiene porque encontrarse en el mismo lugar la mano. El otro problema más significativo es el hecho de que los pacientes que realicen la rehabilitación tienen que tener el brazo siempre en el aire, sin apoyo. Como este proyecto está pensado para personas de todas las edades, este dilema se agranda en personas de tercera edad, con dolencias o dificultades para mantener el brazo en alto Para poder solucionar esto es necesario utilizar un soporte que permita reposar al antebrazo manteniéndolo así siempre en la misma posición, sin que al paciente le resulte un esfuerzo añadido ponerse en él y pudiendo conseguir resultados precisos en cada sesión. Para ello se ha creado un soporte con una impresora 3D que permite mantener el antebrazo en la posición horizontal, paralelo a la mesa donde se apoye.

Este soporte, anterior a este proyecto, sirve para mantener el antebrazo en una posición paralela a la mesa donde se elabore la rehabilitación y es utilizado en los movimientos de la muñeca. En cuanto a los movimientos de los dedos ha sido necesario la elaboración de otro soporte para que el codo esté apoyado en la mesa, pero el resto del antebrazo esté elevado formando este un ángulo con la mesa. Este soporte es provisional y dependiendo del feedback obtenido por los pacientes se puede modificar y por ello se ha hecho con poliuretano el primer modelo del mismo, que es el que se puede observar en la figura 1.6.

Figura 1. 6. Soporte Leap Motion

1.5.1.3. Ordenador

Para crear el videojuego se requiere de un ordenador potente que permita el uso de un desarrollador de videojuegos potente como es Unity, que tiene unos requisitos mínimos para su funcionamiento como desarrollador bastante altos [7]. Entre ellos se exige:

- Sistema Operativo(SO): Windows 7 SP1+, 8, 10; Mac OS X 10.9+.
- GPU: Tarjeta gráfica con DX) (modelo de shadr 3.0) 0 DX11 con capacidades de funciones de nivel 9.3.

Para el desarrollo de plataformas en Windows es necesario adicionalmente:

- SO: mínimo Windows 8.1 (64 bits).
- Visual Studio 2015 o superior.
- SDK de Windows correspondiente.

Es necesario descargarse el software adecuado del *Leap Motion* así como las *Assets* necesarias en función del juego a programar.

En función de la complejidad del juego que se desarrolle puede llegar a ser necesario unas características mayores y una gran cantidad de memoria RAM.

El ordenador escogido para trabajar es el ASUS X550VX que se puede ver en la figura 1.7.

Figura 1. 7. Ordenador ASUS X550VX

Este portátil tiene las siguientes características:

- Windows 10
- Pantalla LED de 15.6"
- Resolución HD 1366x768
- CPU: Intel Core i7 6700HQ (2.6GHz – 3.5GHz)
- Gráficos: -Nvidia GTX 950M
- HDD: 1TB
- RAM: 8GB
- Wi-Fi

Tabla 1. 1. Características Ordenador

1.5.2. SOFTWARE

Esta palabra hace referencia al equipo lógico o soporte lógico de un sistema informático y comprende todos los componentes lógicos necesarios para realizar las tareas específicas [8]. En este caso el software utilizado es Unity y Visual Studio.

1.5.2.1. Unity

Figura 1. 8. Logo Unity [8]

La empresa Unity Technologies es la propietaria del motor de videojuegos Unity [9]. Este motor es utilizado para desarrollar videojuegos para múltiples plataformas como ordenador, smartphones y páginas web. Permite su desarrollo tanto en Windows como OSX y Linux.

Una de las muchas ventajas que tiene esta herramienta es que gracias a un editor y un scripting es capaz de crear videojuegos con un acabado profesional. Existen distintas versiones, con una versión gratuita, y otras de pago con más funcionalidades y menos límites, que es usada por profesionales del sector.

Unity 3D tiene un editor visual muy útil y completo con el cual podemos importar modelos 3D, texturas, sonidos, Scripts, etc. Además, cuenta con un App Store propio en el que podemos obtener muchos modelos gratuitos o de pago con los que realizar nuestro juego.

Se pueden crear videojuegos tanto en 3D como en 2D. Y en la página oficial de Unity3D cuenta con una gran variedad de tutoriales con los que poder iniciarse en el desarrollo de videojuegos

Para crear scripts, que es el código utilizado con la lógica para que realice acciones el videojuego, es necesario tener un compilador externo con el que poder crear o modificar el código. Los idiomas de programación que permite utilizar son JavaScript, C# y un dialecto de Python llamado Boo.

1.5.2.2. VISUAL STUDIO

Figura 1. 9. Logo Visual Studio

Es el compilador que viene integrado en los sistemas operativos Windows. Es necesario un entorno de programación para poder crear los scripts en Unity del *serious game.* En este caso al ser un programa que viene incluido en el paquete de Windows es de muy fácil acceso y existen múltiples tutoriales y documentos en los que obtener información de él.

Para este trabajo se ha utilizado la versión Visual Studio 2015.

2. ESTADO DEL ARTE

Este proyecto consiste en el desarrollo de una aplicación del tipo *serious game* para usarse en la rehabilitación de la operación del síndrome del túnel carpiano. Por tanto, para poder entender este trabajo, es necesario explicar el contexto en el que este se encuentra. Para ello se van a exponer los siguientes apartados: rehabilitación tradicional, *serious games*, sistemas de captura de movimientos y utilización del *Leap Motion* en la actualidad en el ámbito de la medicina.

2.1. REHABILITACIÓN TRADICIONAL

Para poder entender la rehabilitación tradicional hay que comprender el origen de la Fisioterapia y la evolución de la misma [10]. Puesto que es esta ciencia la que se dedica

a tratar la rehabilitación física para mejorar la calidad de vida de los pacientes que han sufrido alguna lesión, ya sea a nivel neurológico o fisiológico.

La palabra fisioterapia proviene de la unión de las palabras griegas physis y therapehia. Que significan naturaleza y tratamiento respectivamente. Es decir, que si nos regimos por su origen etimológico la fisioterapia consiste en "El tratamiento por la Naturaleza". Esto es debido a que en la antigüedad las personas que hacían el papel que hoy hacen los fisioterapeutas utilizaban agentes naturales para el tratamiento que realizaban. Estos tratamientos tienen como fin conseguir que las personas que los reciben puedan realizar de forma independiente las actividades básicas de la vida diaria (ABVD), o si fuera posible, recuperar o disminuir gracias a ciertos ejercicios la función o funciones perdidas, con independencia del motivo de la pérdida.

En la actualidad la Organización Mundial de la Salud (OMS) define la Fisioterapia como *"el arte y la ciencia del tratamiento por medio del ejercicio terapéutico, calor, frío, luz, agua, masaje y electricidad"*. Esta definición nos sirve para entender que, en la historia de la Fisioterapia, se ha utilizado el ejercicio con una concepción terapéutica. Como se puede observar por la definición que da la OMS de la Fisioterapia, esta ciencia hace uso de multitud de elementos para conseguir que las personas recuperen la funcionalidad perdida. Para centrarnos en el ámbito que nos atañe, nos centraremos en el ejercicio realizado por los fisioterapeutas en la rehabilitación del síndrome del túnel carpiano y la utilidad del mismo.

2.1.1. REHABILITACIÓN DEL SÍNDROME DEL TUNEL CARPIANO

El síndrome del túnel carpiano afecta al túnel con el mismo nombre situado en la muñeca [11]. Este canal está delimitado por los huesos del carpo en la parte posterior y el ligamento trasverso del carpo en la parte delantera. Este ligamento comienza en el escafoides y trapecio y se inserta en el gancho y pisiforme. Es decir, recorre todo el brazo hasta llegar a la palma de la mano. El síndrome de este ligamento es una patología que afecta al nervio mediano y se pinza o comprime dentro de este túnel. El lugar donde se encuentra el nervio que se pinza es el que se puede ver en la figura 2.1.

Figura 2. 1. Pinzamiento nervio mediano [11]

Este síndrome es la neuropatía más común del miembro superior, por tanto, es la causa más común del dolor de muñeca.

Esta patología tiene como remedio principal el reposo. Existen muchas otras terapias, que pueden incluir calor y frío o electroestimulación que están siendo utilizadas para tratar esta enfermedad. Con esto se consigue que baje la inflamación producida en esa zona y el dolor. Este tipo de terapia sirve si los síntomas son leves y moderados. Si la sintomatología es grave será necesario un tratamiento quirúrgico. Esta operación consiste en cortar el ligamento carpiano con anestesia local (ver figura 2.2.), dejando una cicatriz pequeña por no ser una operación muy invasiva. El periodo de recuperación depende de cada paciente, pero suele ser menor de un mes.

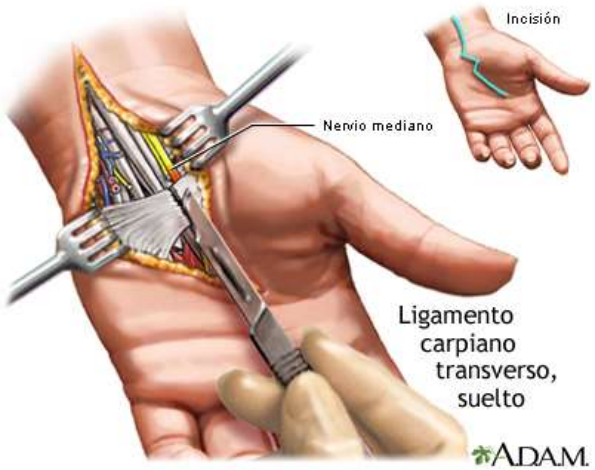


Figura 2. 2. Operación túnel carpiano [11]

Una vez se ha operado, la mano tiene menor movilidad y es necesario otro tipo de terapia. Esta terapia consiste principalmente en forzar al paciente a realizar movimientos que le lleven al límite de la posición que puede tener como máximo en ese momento. Con este tipo de estiramientos se consigue recuperar totalmente el movimiento normal anterior a la patología.

Este tipo de situaciones, en las que hay que realizar el postoperatorio son en las que nos vamos a fijar a la hora de realizar el TFG puesto que está pensado para esa situación el *serious game* realizado. Para poder simplificar en todo este proyecto se hablará de la rehabilitación del túnel carpiano refiriéndose en todo caso a la rehabilitación post operación.

2.2. SERIOUS GAMES

Los denominados *juegos serios* han existido en la sociedad mucho antes de la entrada de los videojuegos en el mundo del entretenimiento [12]. Pero estos *serious games* han experimentado un crecimiento exponencial con la llegada de los dispositivos informáticos y electrónicos. Su avance, en cierta medida, depende del avance de los videojuegos comunes puesto que se nutren de ellos para su creación y desarrollo. Los juegos que son denominados así son los que tienen una finalidad distinta a la de entretener. Esto no significa que no tengan a su vez un carácter lúdico y han sido ampliamente utilizados dentro de los ámbitos de la educación, sanidad, política pública y comunicación estratégica entre otros.

Dentro del sector de la sanidad ya existen incluso cursos como es el *Serious Games For Health* que se impartió por primera vez en Barcelona en abril del 2016 y que ha dado lugar a la creación del libro *Serious Games For Health*. *Mejora tu salud jugando*. Este libro está orientado al sector sanitario y, con este, pretenden generar "una oportunidad sin precedentes" con la que motivar y cambiar la conducta de las personas respecto a sus enfermedades y dolencias. El objetivo del libro, por tanto, es concienciar

para adaptar el diseño de juegos desde una orientación sanitaria. Este a su vez es uno de los objetivos que tiene el proyecto del que se está realizando la memoria.

Para comprender la gran aceptación que estos tienen conviene hablar de varios ejemplos que son utilizados en distintos ámbitos con los que poder explicar su utilidad.

2.2.1. PULSE!!

Dentro del sector médico, aparte de utilizar este tipo de juegos con fines terapéuticos, también se está utilizando con fines educativos [13]. Gracias a la experta en enfermería Claudia Johnston se ha desarrollado el juego *Pulse!!* Dicho juego se basa en los videojuegos de disparos en primera persona para reproducir las condiciones de una sala de emergencias en un hospital como se puede ver en la figura 2.3.

Figura 2. 3. Juego Pulse!! [13]

Gracias a este juego, los futuros enfermeros pueden practicar lo que han aprendido en clases teóricas y así ganar experiencia enfrentándose a experiencias reales sin el riesgo que la inexperiencia en ellas conlleva. Este tipo de aplicaciones permiten a los jugadores identificar los problemas de cada paciente para así poder priorizar a los más leves y aplicar las medidas necesarias para su sanación.

2.2.2. SIMULADOR FORMULA 1

Dentro del espectáculo que es la Formula 1 cada vez es más utilizada la realidad virtual para diseñar y probar sus monoplazas [14]. En este deporte solo se tienen 10 días de entrenamientos en pista durante la preparación del campeonato y es por ello que los equipos se han visto obligados a utilizar los *serious games* para probar sus innovaciones.

Este es el caso de la Scuderia Ferrari, que en 2010 diseñó uno de los simuladores más avanzados hasta ese momento, como se puede ver en la figura 2.4.

Figura 2. 4. Simulador de F1 [14]

En esta ocasión se utiliza este simulador tanto para que el piloto se adapte a los cambios que tiene el coche y a cada circuito pudiendo probarlo en el simulador sin necesidad de utilizar el monoplaza, como para que los ingenieros modifiquen el coche y comprobar las mejoras. Este es uno de los mayores avances que hay en la actualidad dentro del mundo de los *serious games* y, aunque aún no es fiel a la realizad, cada día se avanza más en esa búsqueda de una realidad virtual que no sea posible diferenciar de la realidad. Gracias a estos avances, en los cuales se invierte una gran cantidad de dinero, es posible avanzar en los *serious games* puesto que en otros sectores no están tan

avanzados y la imitación del Simulador de Formula 1 permite un avance mayor en ese campo.

2.3. SISTEMAS DE CAPTURA DE MOVIMIENTOS

A la hora de realizar la rehabilitación es necesario tener algún sistema de captura de movimientos para poder cuantificar los movimientos obtenidos y así corregir o aplicar cambios que puedan mejorar la terapia. Esta necesidad ha sido un tema activo en la investigación desde los años 80, provocado, en gran medida, por el aumento en los pacientes que han sufrido alguna discapacidad motora.

Si se quiere hablar de ese tipo de sistemas conviene nombrar a Johansson y su famoso experimento Moving Light Display (MLD) de 1973. Este experimento consistió en acoplar marcadores reflectantes en distintas articulaciones del cuerpo humano con las que poder percibir el movimiento biológico. Gracias a estos marcadores, y con una luz que conseguía la reflexión de los mismo, se consiguió una monitorización de los mismos y con ello se consiguió un gran avance en el rastreo del movimiento humano.

Tras este experimento, que supuso un gran avance en los sistemas de captura de movimientos, han aparecido multitud de técnicas de captura. Estas técnicas, de diversa índole, van desde trajes mecánicos, sensores de movimiento, cámaras a sensores de sonido. Las ventajas y desventajas de cada uno de los métodos de captura dependen, en gran medida, del presupuesto y de la precisión y realismo que se pretenda conseguir. Estos sistemas de captura se podrían dividir en visuales y no visuales.

Los sistemas ópticos o visuales hacen uso de una o más cámaras para capturar la información y mediante algoritmos programáticos sobreponer las distintas proyecciones del objeto obtenidas para obtener la representación real del objeto.

El uso de materiales reflectivos con marcadores, como en el experimento de Johansson, son, hoy en día, muy utilizados para capturar la posición del cuerpo humano

dando lugar a información que programas como los de la figura 2.5 utilizan para recrear la posición de la persona.

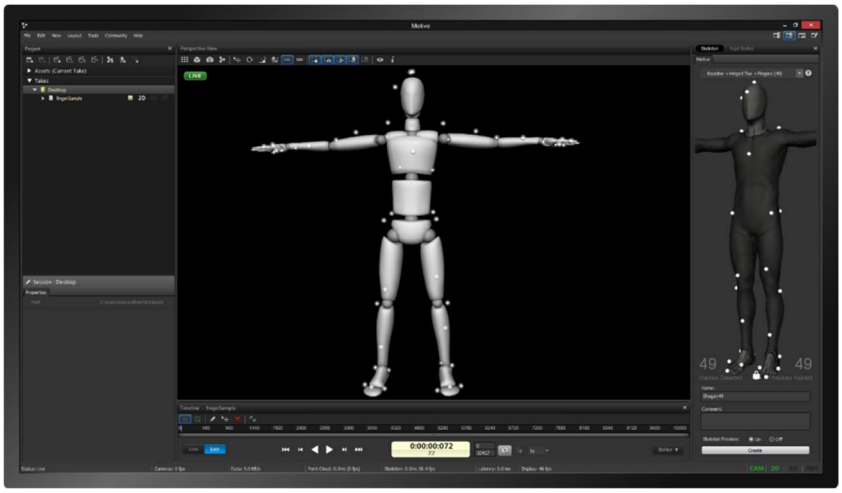


Figura 2. 5. Cuerpo humano digitalizado [15]

Dentro de los sistemas no visuales cabe destacar los sistemas de inercia. Estos sistemas consisten en el uso de acelerómetros o giróscopos en los lugares donde se pondrían los marcadores reflectantes para conocer la posición de la persona gracias a la información que envían de forma inalámbrica. Un claro ejemplo de sistema de inercia es el traje mecánico *Motion Capture* o *MoCap* de la figura 2.6., que cuesta alrededor de 70.000€ [15]. Con este traje se consigue una gran precisión y realismo, más su precio y el hecho de que sea utilizado para capturar los movimientos de todo el cuerpo y no de una parte precisa pueden hacer de esta elección la incorrecta en función de la situación.

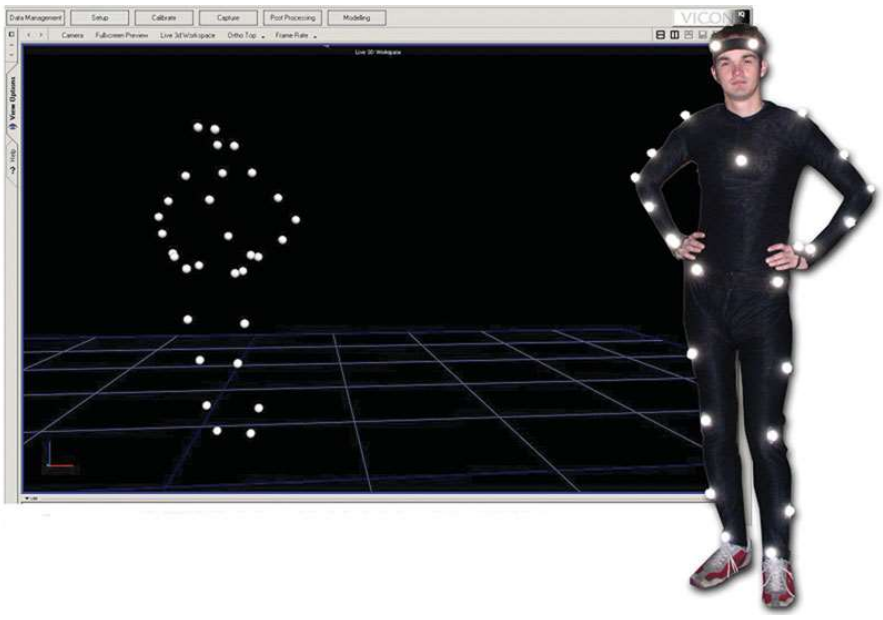


Figura 2. 6. Traje MoCap

2.4. UTILIZACIÓN DEL LEAP MOTION EN LA MEDICINA

El dispositivo *Leap Motion* está siendo utilizado con una gran variedad de aplicaciones dentro del campo de la medicina. Esto se debe a la evolución que ha significado tener un dispositivo que es capaz de captar la posición de las manos en todo momento y de transmitir gran cantidad de información que puede ser analizada. Esto ha provocado que esté siendo utilizado en varios campos de la medicina como son la terapia física, investigación hospitalaria para entrenamiento quirúrgico o para ayudar a personas con necesidades especiales. A su vez, gracias a módulos integrados innovadores se está consiguiendo que cirujanos tengan acceso a información médica vital dentro de un ambiente esterilizado sin necesidad de salir de él.

Las principales formas en las cuales está siendo utilizado el *Leap Motion* dentro de la medicina [16] son las siguientes:

1. Para interpretar el lenguaje de signos. El *Leap Motion* tiene gran utilidad para poder mantener una comunicación bidireccional entre una persona que utiliza el

lenguaje de signos y otra que no. La revista *TIME* publicó la invención de *MotionSavy´s UNI*. Este software permite que personas sordas se comuniquen gracias a la traducción del lenguaje de signos por medio del *Leap Motion* y traducción posterior al lenguaje escrito o hablado gracias a tecnologías de voz como se puede ver en la Figura 2.7.

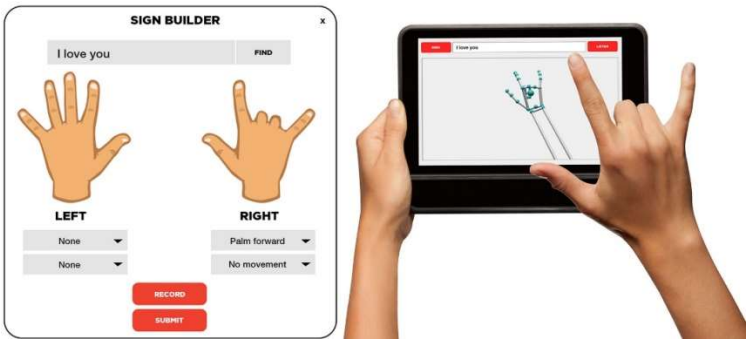


Figura 2. 7. MotionSavy´s

2. Para mejorar la visión. Ciertas alteraciones en el funcionamiento del ojo humano como son el estrabismo o los ojos vagos son provocadas cuando el cerebro humano ignora cierta información que les llega del ojo no dominante. Gracias a la empresa *Vivid Vision* se han creado juegos de realidad virtual con los que forzar a ese ojo no dominante a ser el más utilizado forzando al cerebro a nivelar la información que recibe de cada ojo.

Figura 2. 8. Utilización software Vivid Visión

3. Para cuantificar los temblores de manos. Ciertas enfermedades como son la enfermedad de Wilson o el Parkinson provocan en las personas que las padecen temblor de manos que aumenta con el tiempo. Gracias a la cantidad de información que recibimos del *Leap Motion*, hay investigadores que están desarrollando aplicaciones para poder medir ese temblor y así comprobar la progresión en el tiempo de forma cuantitativa de dichos movimientos.

4. Para la rehabilitación de personas con daños en la médula espinal. Dependiendo de la lesión que se tenga el cuerpo humano puede presentar plegias que son reparables o al menos con cierto nivel de evolución. En el Hospital de Parapléjicos de Toledo se está investigando el uso del dispositivo *Leap Motion* para realizar la rehabilitación de los miembros superiores del cuerpo.

5. Para manejar documentación médica en salas esterilizadas. Las salas de operaciones son necesarias que estén esterilizadas para realizar operaciones en ellas. En ellas se maneja documentación médica, la cual no puede ser modificada en un ordenador porque al manejar el ratón puede producirse una contaminación cruzada. La empresa *TedCas* está desarrollando un software con el cual poder administrar esa información sin utilizar los periféricos convencionales del ordenador gracias a movimientos intuitivos en el *Leap Motion*.

6. Para la rehabilitación física y análisis del estado actual. Anterior a este trabajo existe otro trabajo realizado por la Ingeniera Cynthia Cubeiro para el proyecto *RoboHealth* consistente en la simulación de un piano de 10 teclas en el cual realizar movimientos de los distintos dedos de la mano para comprobar el tiempo y evolución en los movimientos de tocar cada una de las teclas con cada dedo de la mano.

3. DESARROLLO DEL TFG

Con el fin de conseguir desarrollar un *serious game* se han ido implementando una serie de procesos con los que llegar al objetivo final de crear un sistema de rehabilitación para el síndrome del túnel carpiano. Para ello lo primero que se ha hecho es desarrollar varios videojuegos, sin utilizar el *Leap Motion*, con los que familiarizarse con el entorno de programación de Unity. Posteriormente se ha utilizado el dispositivo de detección de manos para comprobar las métricas que es capaz de recoger Unity del *Leap Motion*. A su vez, tras crear este sistema de evaluación que guarda las métricas obtenidas de los ángulos máximos obtenidos por los movimientos de muñeca y de las distintas falanges de los dedos se ha comprobado al fiabilidad de los resultados y la precisión de los mismos puesto que este tipo de terapia que cuantifica la posición en la que se encuentra la mano es novedosa y no existe en el sistema actual ningún estudio de referencia a partir del cual poder despreciar o considerar los datos que se obtienen. Por último, se ha creado una interfaz sencilla que pueda ser utilizada por los pacientes o los médicos y dos videojuegos en los que, gracias al uso del *Leap Motion* y a las métricas obtenidas en el sistema de evaluación, permite al usuario realizar movimientos para rehabilitar el túnel carpiano. A su vez se ha incluido una leve explicación del funcionamiento de cada una de las opciones en cada juego.

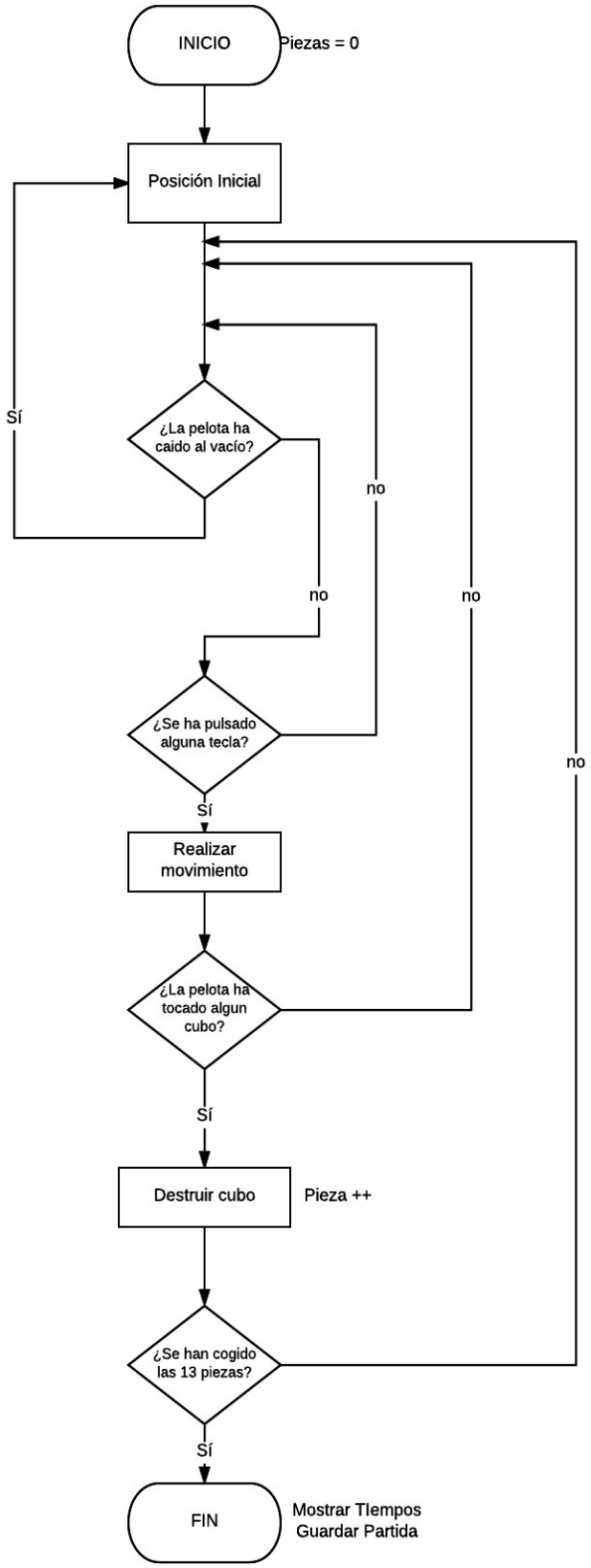
3.1. VIDEOJUEGOS INICIALES

Para poder comprender el funcionamiento del entorno gráfico de *Unity3D* se han desarrollado una serie de juegos con los que adaptarse a la forma de programar de esta aplicación y entender así cómo continuar en la elaboración del TFG [17].

3.1.1. ROLL A BALL

Este videojuego consiste en una pelota que se mueve cuando se toca alguna de las flechas del teclado moviéndose en la dirección pulsada siguiendo la cámara en todo momento a la pelota. A su vez si se pulsa la tecla espacio salta y si cae al vacío la pelota vuelve a la posición de inicio. El juego consta de 13 cubos que hay de coger para ganar y

se guarda en un archivo el nombre de la persona que ha realizado el juego y el tiempo que ha tardado. Todo esto queda más claro en el siguiente flujograma [18]:



Flujograma 3. 1. Juego Roll a Ball

Una vez creado el primer juego y programados los scripts necesarios para su funcionamiento se diseñaron 3 niveles con los que aumenta su dificultad como se puede ver en las siguientes figuras.

En la figura 3.1. solo es necesario manejar las teclas para pasarse el juego.

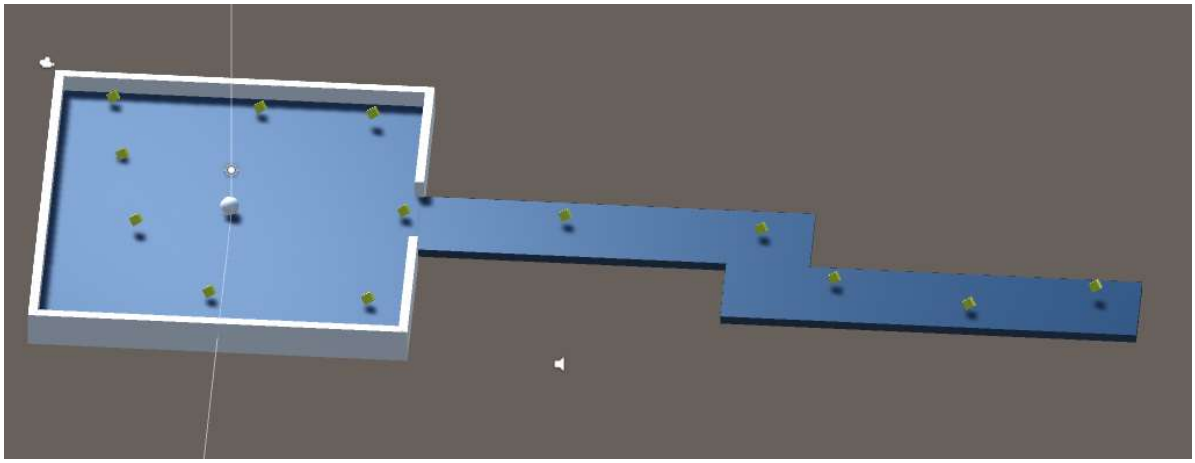


Figura 3. 1. Roll a Ball Nivel 1

En el segundo nivel se ha añadido un desnivel que para superarlo hay que usar la barra espaciadora para saltar.

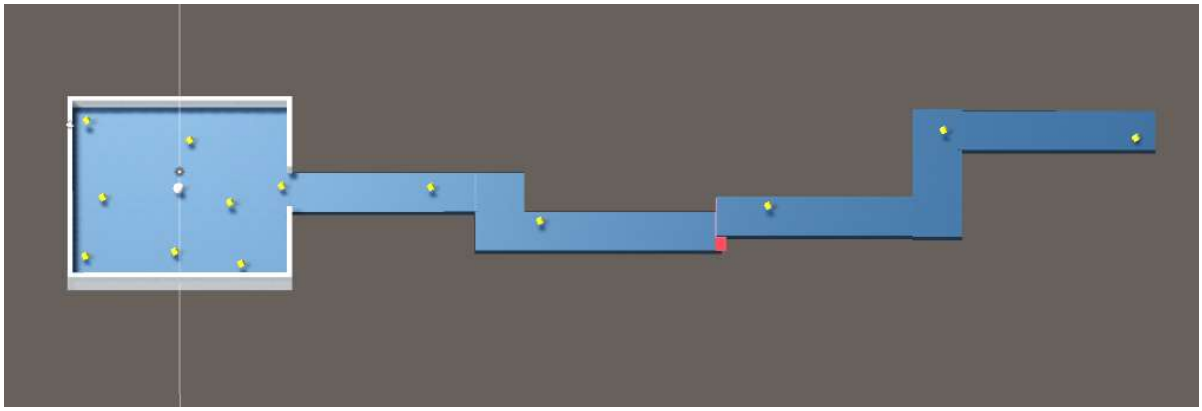


Figura 3. 2. Roll a Ball Nivel 2

Por último, en el tercer nivel, que es el que se puede ver en la figura 3.3. se puede ver como la dificultad aumenta añadiendo más saltos y plataformas.

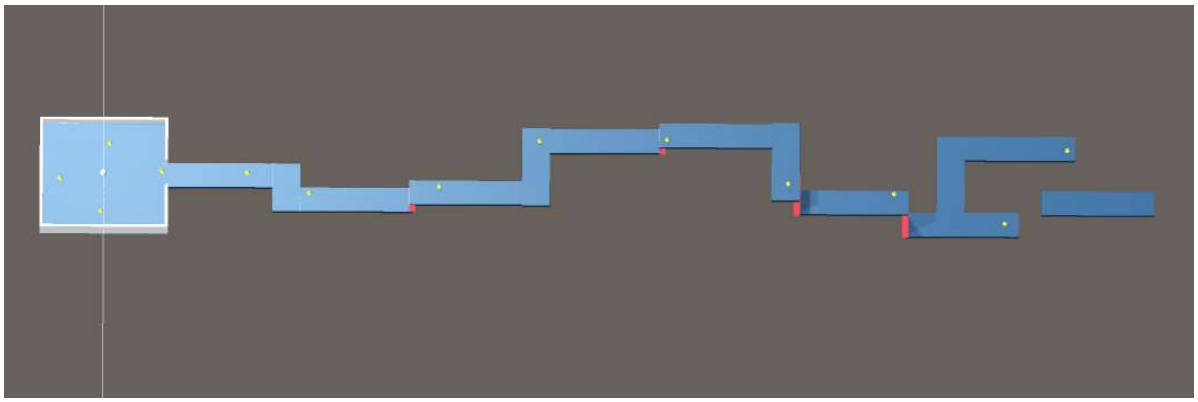


Figura 3. 3. Roll a Ball Nivel 3

3.1.2. PLATAFORMAS 2D

Este segundo videojuego utiliza scripts ya creados y modelos 3D para desarrollar un juego de plataformas en 2 Dimensiones. En este caso la forma de manejar al personaje es pulsar las flechas de izquierda y derecha para moverse y la barra espaciadora si se quiere saltar. Todas las partes de este juego han sido extraídas de las librerías que ya tiene Unity3D y en concreto se ha importado los Assets de 2D.El jugador es importado de la librería (ver figura 3.4.). Todo esto ha servido para conocer como funcionan las librerías en Unity y como de ellas sin programar o modificando los scripts se puede crear un videojuego. A su vez ha sido útil para crear la interfaz posterior en la que se basa el *serious game* final.

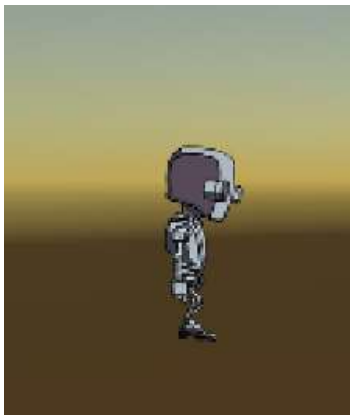


Figura 3. 4. Personaje Plataformas 2D

Como en el Roll a Ball hay que llegar a un objetivo final, que en este caso es que el personaje entre en contacto con el tesoro final, que indicará que has ganado, y guardará el tiempo que se ha tardado en llegar en otro archivo junto al nombre del usuario. El escenario es el que se puede ver en la figura 3.5.

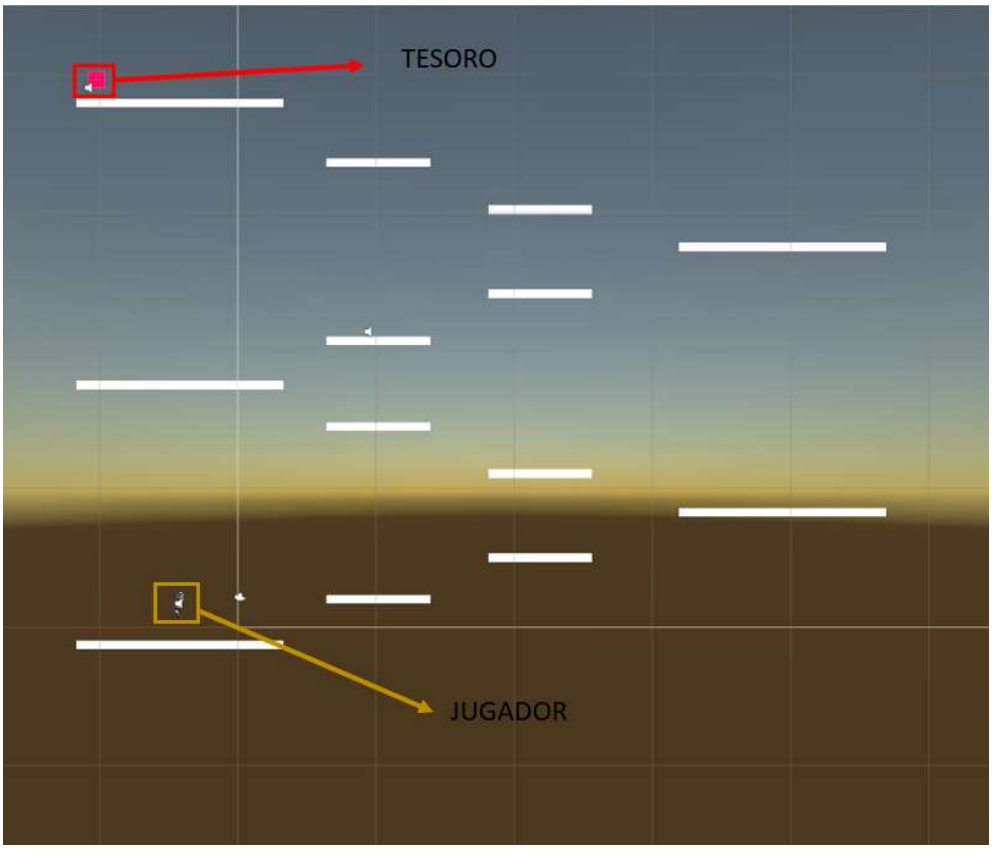


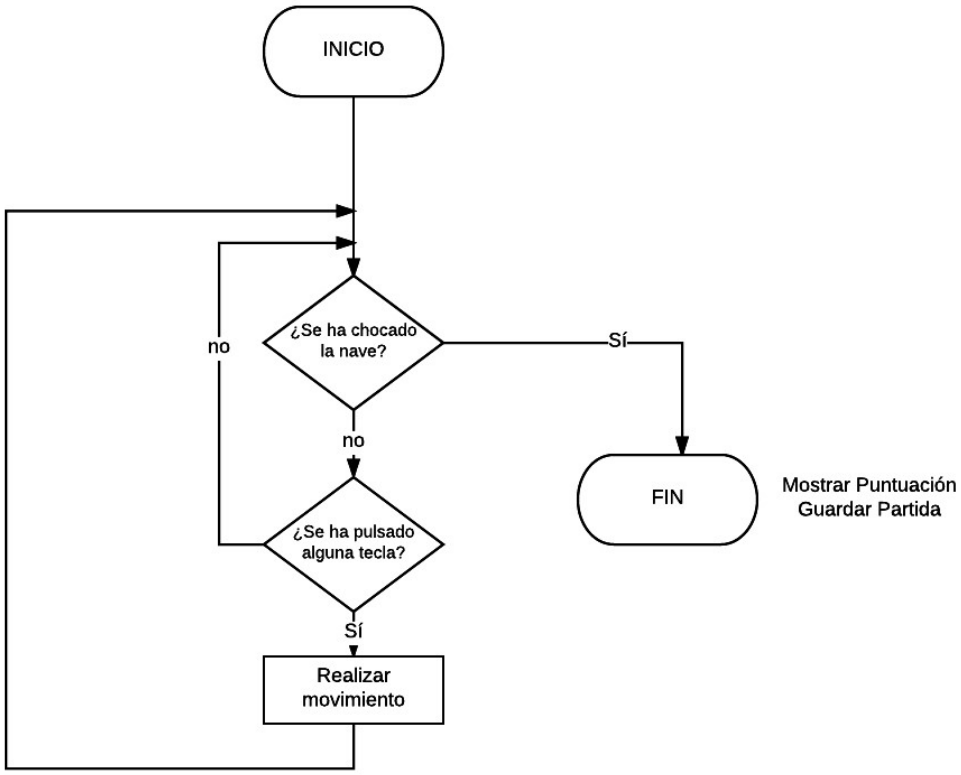
Figura 3. 5. Escenario Plataformas 2D

3.1.3. SPACE SHOOTER

Este tercer y último videojuego creado para entender cómo funciona Unity se basa en el juego clásico de naves Space Shooter. En este caso se ha creado un escenario en el cual una nave, cuyo modelo 3D ha sido importado desde el mercado de Unity, se mueve al pulsar las teclas de flechas y dispara al pulsar el botón izquierdo ratón para destruir los asteroides que se acercan a la nave. El escenario en el que se mueve el juego es el de la figura 3.6.

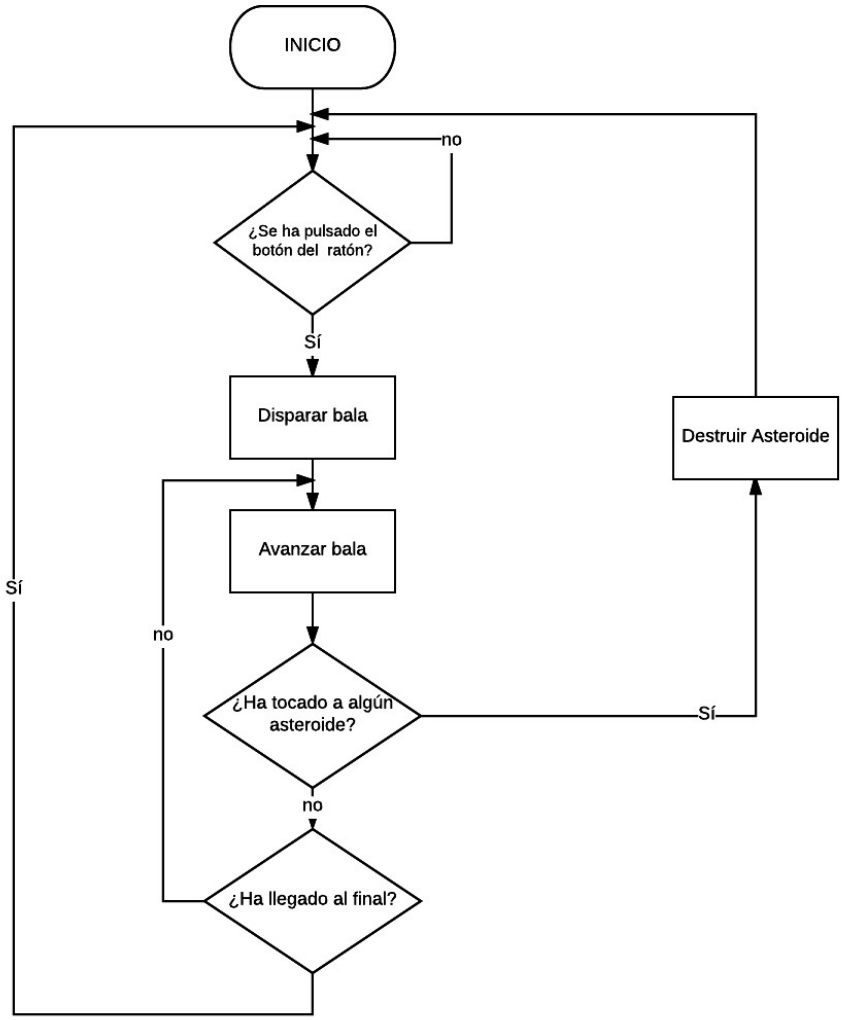
Figura 3. 6. Naves y Asteroides

Los asteroides van llegando por hordas y si alguno entra en contacto con la nave, esta se destruye dando por finalizado el juego y dejando un archivo de Excel con los datos del tiempo que se ha aguantado, las horas que se han destruido y la puntuación obtenida por destruir asteroides, así como el nombre del jugador. Todo esto queda mejor explicado en el siguiente flujograma en el que se explica el movimiento de la nave.



Flujograma 3. 2. Movimiento Nave

La lógica que sigue la bala para destruir un asteroide es la que se puede ver en el siguiente flujograma.



Flujograma 3. 3. Movimiento Bala

3.2. SISTEMA DE EVALUACIÓN

El siguiente paso en el desarrollo del TFG para conseguir los objetivos marcados es la utilización del dispositivo *Leap Motion* dentro del motor de Unity. Para ello es necesario descargarse de la página oficial de L*eap Motion* el conjunto de *Assets Orion* Beta para *Leap Motion*. Una vez descargado el SDK se puede ya trabajar con el *Leap* dentro de *Unity3D* puesto que viene ya con algunos ejemplos para *Unity*. A su vez existen distintos módulos que son integrables en Unity para implementar ciertas funciones en

Unity que no son necesarias en nuestro caso, como es el *Leap Motion Interaction Engine* que sirve para detectar colisiones de la mano con objetos en Unity, amén de ejemplos y tutoriales para comprender su uso.

Para el sistema de evaluación se pueden distinguir dos apartados, que son las escenas en las que se ha realizado la captura de los ángulos de la palma de la mano y de las distintas falanges de los dedos respectivamente. En ambos casos se ha realizado solo el cálculo del ángulo de la mano derecha como prueba inicial.

3.2.1. EVALUACIÓN MANOS

Este primer sistema de evaluación tiene como objetivo capturar el ángulo máximo y mínimo que es capaz de realizar el paciente en una serie de movimientos [19].

- Flexión: dentro del sistema de evaluación se ha llamado Vertical Min, siendo un ángulo negativo el resultado obtenido. Este movimiento consiste en acercar las caras posteriores del antebrazo y la mano inclinando la palma hacia abajo.

- Extensión: en el entorno de programación se ha llamado Vertical Max y el resultado de realizar este movimiento da un ángulo positivo. La extensión se da cuando se acercan las caras posteriores del antebrazo y de la mano, inclinando la palma hacia arriba. Estos movimientos se pueden ver en la figura 3.7.

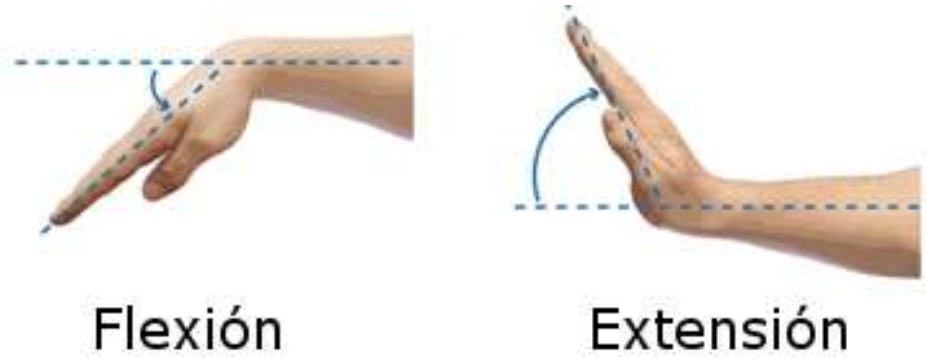


Figura 3. 7. Flexión y Extensión

- Abducción: este movimiento también llamado inclinación radial consiste en acercar los bordes externos del antebrazo y de la mano, es decir, es el movimiento lateral que acerca el dedo pulgar al antebrazo. Dentro del *serious game* se ha llamado ángulo Horizontal Min al valor en negativo del ángulo que se obtiene respecto a poner la palma en línea con el antebrazo. Es la nombrada como B dentro de la figura 3.8.

- Aducción: esta última posición que adquiere la mano también es denominada inclinación cubital. Consiste en un desplazamiento lateral que acerca los bordes internos del antebrazo y de la mano, realizando el movimiento contario a la abducción. Se ha llamado ángulo Horizontal Max al ángulo positivo que se obtiene de calcular el ángulo que la palma tiene respecto a la palma en línea con el antebrazo. Es la posición A en la figura 3.8.

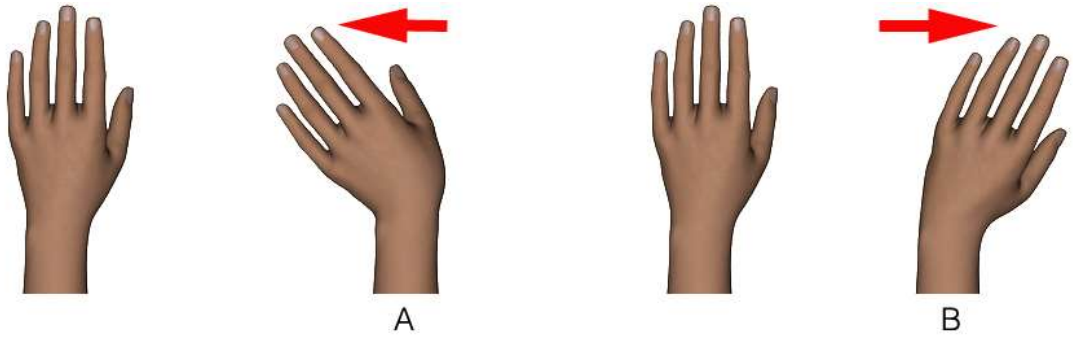


Figura 3. 8. Abducción y Aducción

La forma de proceder con la evaluación de la mano consiste en realizar primero el movimiento de flexión y extensión y una vez han quedado claros el ángulo máximo y mínimo al que puede llegar el paciente hacer lo propio con los movimientos de abducción y aducción. Para ello se ha creado una interfaz por la cual se muestra en todo momento el valor máximo y mínimo del ángulo conseguido en cada movimiento como se observa en la figura 3.9.

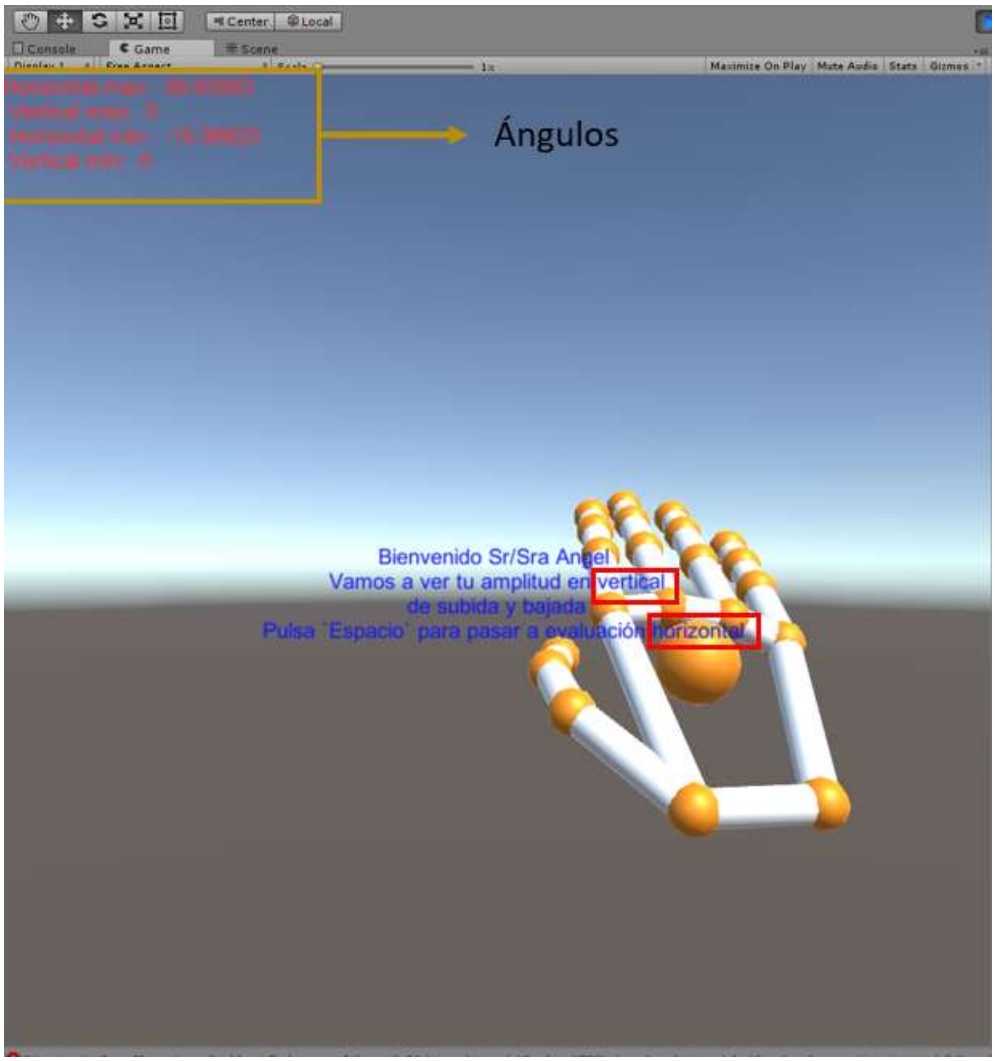
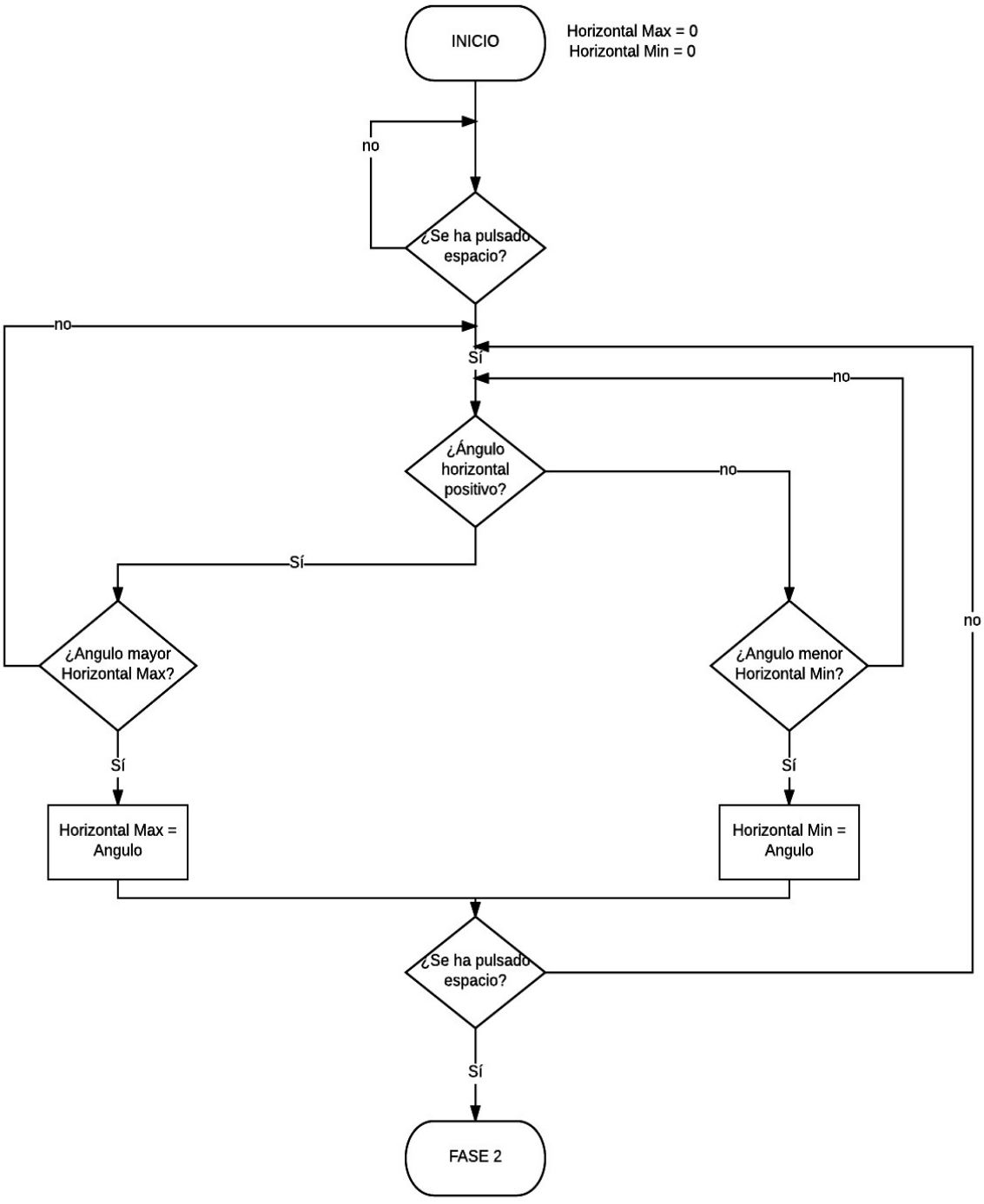


Figura 3. 9. Movimiento vertical

Una vez se ha acabado la evaluación da la posibilidad de guardar las métricas obtenidas o reiniciarlo otra vez con el texto que se puede ver en la figura 3.10.

Figura 3. 10. Texto Final Mano

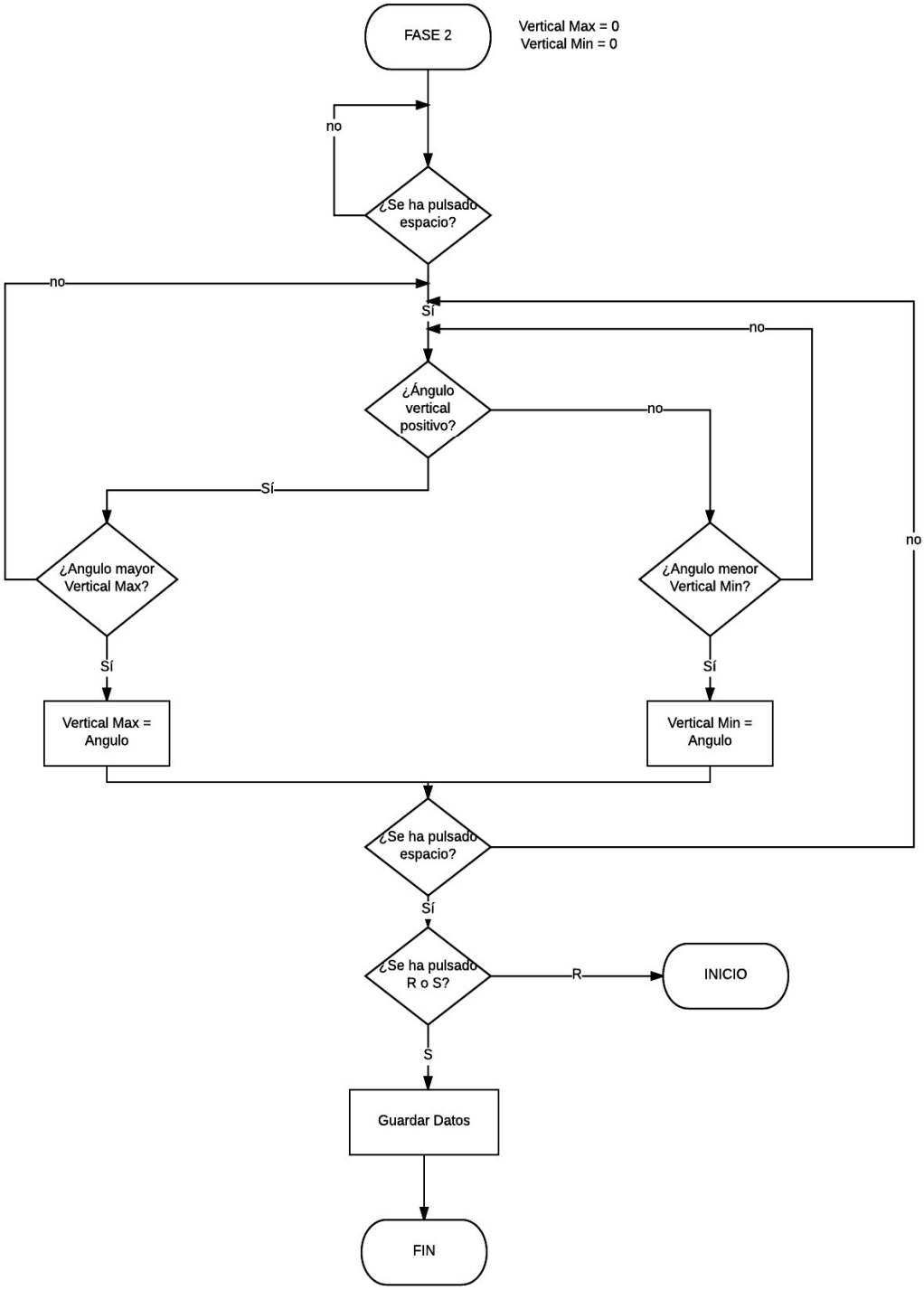
Para conseguir que guarde en todo momento el valor del ángulo máximo al que puede llegar se han tenido que crear varios scripts así como la lógica necesaria para que funcione, que es la que se puede ver en el siguiente flujograma.



Flujograma 3. 4. Cálculo ángulo Flexión Extensión

El ángulo inicial que tiene máximo y mínimo es 0. Mientras esté en la fase de detección del ángulo de flexión y extensión comprueba constantemente si el ángulo que forma la palma de la mano con el suelo (considerando suelo una superficie sin inclinación) es mayor que el ángulo que se tiene en ese momento como ángulo máximo o mínimo. Si es mayor se guarda ese nuevo ángulo como Horizontal Max u Horizontal Min. Con esta lógica se consigue que siempre quede guardado el ángulo máximo y mínimo que consigue el paciente con ese movimiento cuantificando así un gesto que los fisioterapeutas no pueden hacer con precisión.

A la hora de hacer el cálculo de Vertical Min y Vertical Max se procede de la misma forma solo que cambiando lo que consideramos que es la posición que es el ángulo 0, que antes era el suelo para calcular Vertical Max y Vertical Min. Una vez finalizado se guarda en un archivo las métricas obtenidas junto con el nombre del paciente como se puede ver en el siguiente flujograma.



Flujograma 3. 5. Calculo ángulo Abducción y Aducción

3.2.2. EVALUACIÓN DEDOS

A la hora de conseguir cuantificar el movimiento de los dedos para poder analizar posteriormente los datos obtenidos se ha seguido una estructura parecida a la seguida con la mano. Se ha ido calculando en todo momento cual es el ángulo máximo que forman las falanges proximales y medias de cada dedo. Este ángulo calculado se ha hecho siempre respecto al suelo como se hizo en la Flexión y Extensión, mas en este caso solo se han guardado los datos de flexión. La forma de proceder es la misma y como en el anterior caso se guardan los datos en un archivo junto al nombre del paciente. Los cambios que se producen son en la cantidad de datos a manejar pues habrá que guardar los ángulos de todos los dedos y de las falanges proximales y medias como se puede ver en la siguiente figura.

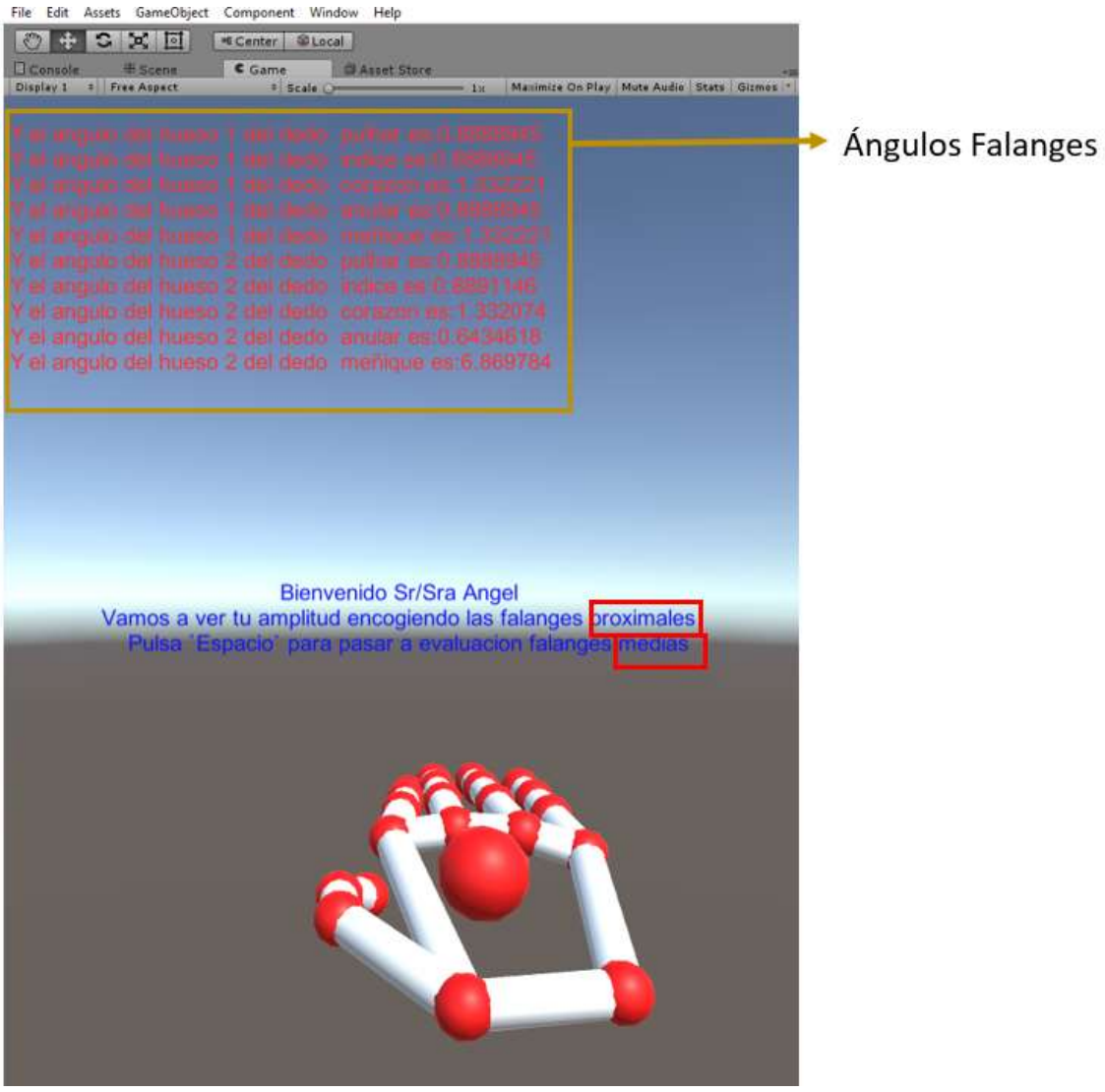


Figura 3. 11. Ángulos Dedos

Posteriormente se ha mejorado el aspecto gráfico del juego para conseguir que el paciente tenga una interfaz más intuitiva y se le haga el juego más ameno como se puede ver en la figura 3.12.

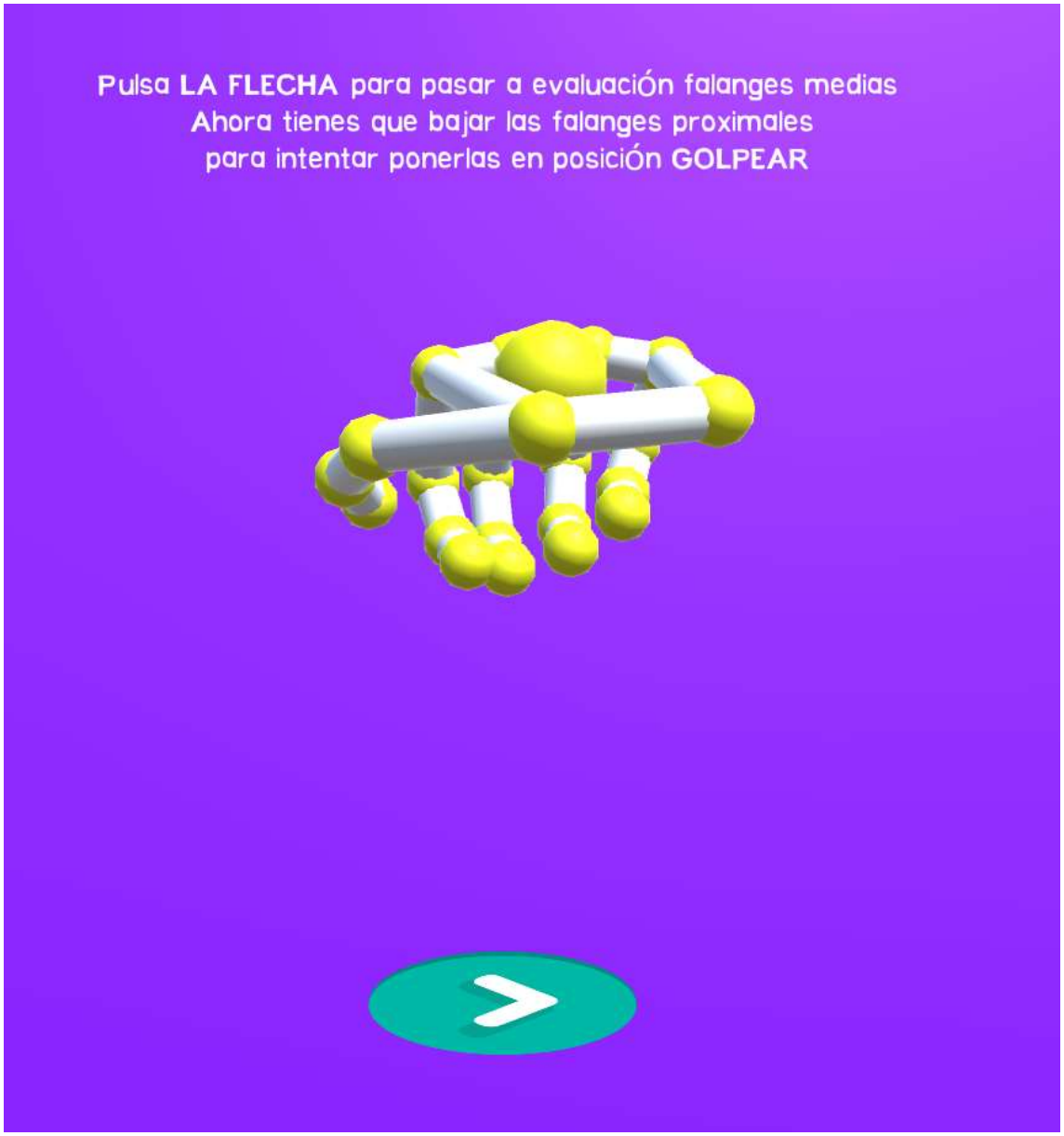


Figura 3. 12. Evaluación Dedos 2

3.3. SERIOUS GAMES

Una vez se han conseguido guardar los datos del ángulo máximo que tiene la palma de la mano y las falanges de los dedos se van a utilizar estos datos para crear dos *serious games* con los que conseguir que el paciente ejercite la zona que repercute en el túnel carpiano. Para ello las máximas que tienen que seguir estos juegos es que tienen que ser sencillos, puesto que si es un juego muy complicado el paciente se va a estresar y no le va a gustar. También tienen que ser juegos aptos para todos los públicos puesto que esta enfermedad se puede producir en personas de cualquier edad. Con el objetivo de obtener un feedback positivo por parte de la persona que juega se han creado dos juegos: Space Shooter 2.0 y Juego Granja. Cada uno de ellos basado en las distintas evaluaciones obtenidas anteriormente.

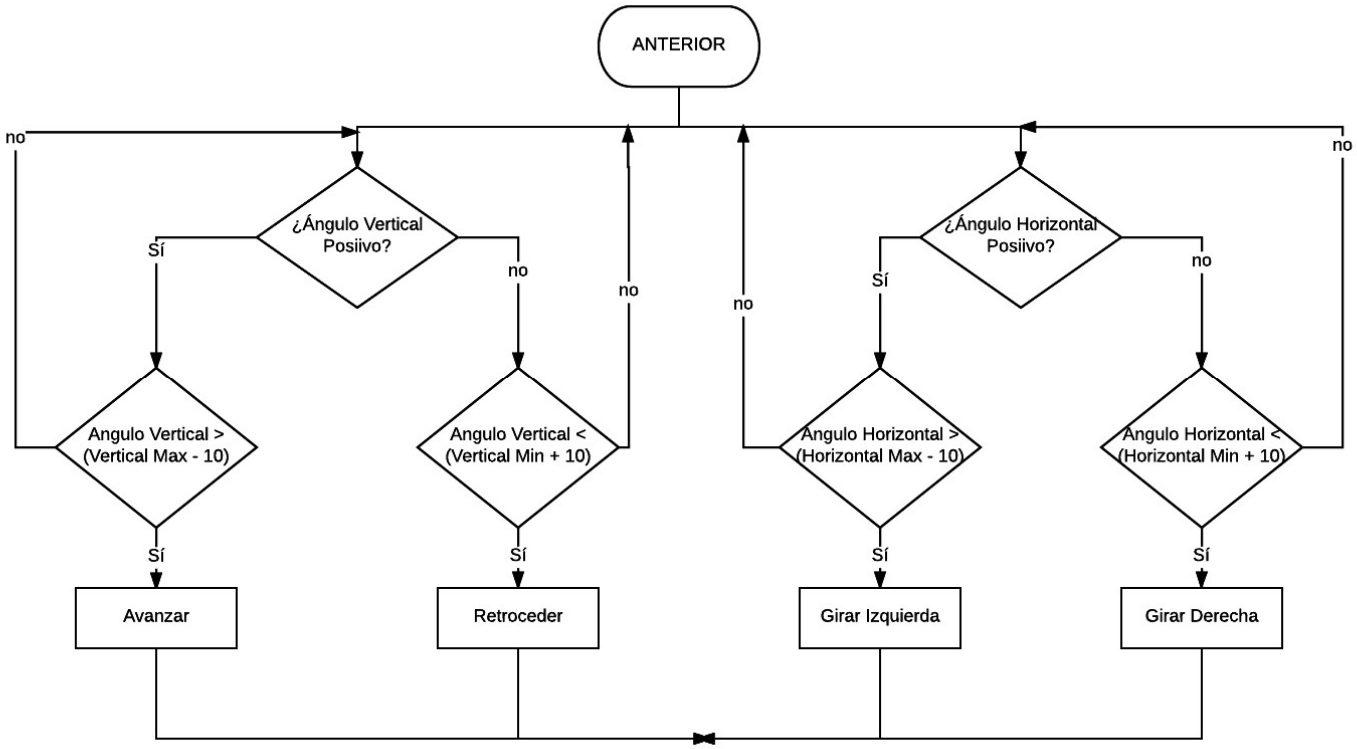
3.3.1. SPACE SHOOTER 2.0

Para poder jugar a ese *serious game* es necesario haber hecho anteriormente la Evaluación Manos puesto que necesita de los datos del ángulo máximo y mínimo de la muñeca para funcionar. En este caso se ha aprovechado el videojuego anterior "Space Shooter" y se han cambiado los controles para mover la nave con la mano. Se han dejado el mismo aspecto visual y efectos que en el anterior juego como se puede ver en la figura 3.13.

Figura 3. 13. Game Over Space Shooter 2.0

Gracias a esto se consigue demostrar que no siempre es necesario la creación de un videojuego nuevo con el que conseguir un *serious game*, sino que adaptando juegos ya existentes se puede conseguir dicho objetivo haciendo que haya juegos ya creados que con una adaptación se pueda conseguir hacer de ellos juegos serios.

En el caso que nos importa el cambio que se ha realizado consiste en cambiar la forma que tiene de mover y disparar la nave. Al añadir el *Leap Motion* al juego se puede capturar el ángulo que tiene la palma de la mano en todo momento. Esos datos obtenidos son comparados con el ángulo máximo y mínimo que tiene el paciente para hacer que se mueva el vehículo. Todo esto se puede ver de forma más clara en el siguiente flujograma.



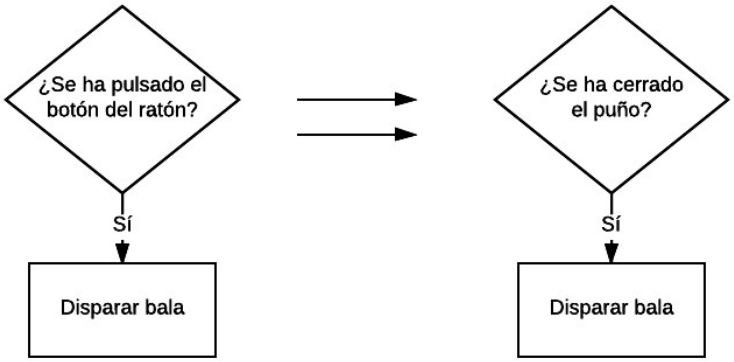
Flujograma 3. 6. Movimiento Nave con la mano

En este caso, como se puede ver en el flujograma la nave se moverá hacia delante si el ángulo que tiene la mano de extensión es mayor que el ángulo máximo de extensión que tiene menos 10 grados. Lo mismo pasa con el resto de movimientos con la variación de 10º respecto al máximo que ha conseguido el paciente haciendo que el movimiento de extensión haga retroceder a la nave, abducción haga girar a la derecha a la nave y aducción la haga girar a la izquierda.

En este juego se ha sustituido la siguiente parte del flujograma del Space Shooter inicial por la lógica aquí explicada.

Flujograma 3. 7. Lógica Space Shooter Inicial

En cuanto a la forma de disparar se ha optado por coger el movimiento de cerrar el puño como acción de disparar. En este caso el cambio en el flujograma es el siguiente.



Flujograma 3. 8. Lógica Disparar

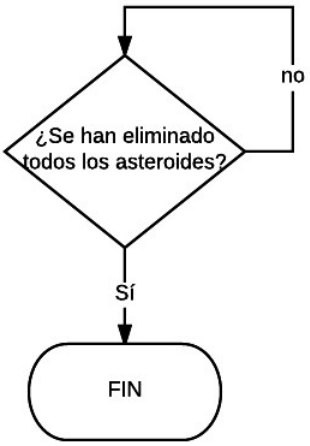
Los movimientos de la mano escogidos para que se mueva la mano se han hecho en base a ser los más intuitivos posibles. De tal manera que la nave en cierto modo imita el movimiento de la mano. En el caso del disparo se ha escogido cerrar el puño pues es un movimiento en el que se ve implicado el túnel carpiano y no se puede realizar el movimiento de disparar de forma intuitiva con la mano si se quiere a su vez manejar la nave.

También se ha cambiado la forma de finalizar el juego. En la anterior versión el juego terminaba cuando la nave era destruida por un asteroide. Para que esto no genere ansiedad o malestar en el paciente se ha cambiado esto. Ahora cada vez que la nave choca

con un asteroide aumenta un contador de fallos del paciente, pero puede seguir jugando como se puede ver en el siguiente flujograma.

Flujograma 3. 9. Nave colisión asteroide

Con lo cual, ahora para terminar el juego hay que eliminar todos los asteroides, y en esta ocasión en cada horda no vuelven a aparecer todos los asteroides, si no que desaparecen tantas rocas espaciales como se hayan destruido en la anterior ronda. Por eso la forma de finalizar es la que se puede ver en el siguiente flujograma.



Flujograma 3. 10. Fin Juego Naves

A su vez, otro aspecto a destacar, es que para conseguir adaptar este juego a todos los públicos se han introducido una serie de variables que permiten modificar el funcionamiento del mismo como son el número de asteroides y la velocidad de los mismos para así hacer que se adapte el juego al rango de movimientos que tiene en ese momento la persona que está interactuando con la nave como se puede ver en la siguiente figura.

Figura 3. 14. Modificaciones dificultad

3.3.2. JUEGO GRANJA

El otro juego del que consta el proyecto se ha llamado juego granja. En este caso se utilizan las métricas guardadas en la evaluación dedos.

Dentro del síndrome del túnel carpiano existen ciertos movimientos y posiciones de las manos que son más útiles para conseguir una rápida mejoría en la parte dañada. Entre estos movimientos están los de doblar las falanges proximales formando un ángulo de 90º entre la palma y los dedos si es posible como se puede ver en la Imagen 3.1. A este movimiento lo llamaremos “Golpear”.

Imagen 3. 1. Posición mano falanges proximales

Otro de los movimientos más importantes es el de doblar las falanges medias en busca de obtener también un ángulo de 90º con la palma, pero en este caso con las siguientes falanges como se puede ver en la Imagen 3.2. A esta posición la llamaremos "Coger".

Imagen 3. 2. Posición mano falanges medias

Para el juego de la granja se han utilizado estos dos movimientos y el de extender la mano como puntos en los que la mano, gracias al *Leap Motion*, interactúa con el juego. Es necesario haber realizado antes la Evaluación Dedos para jugar a este entretenimiento. Dentro de esa evaluación se cogen las métricas necesarias para detectar cuando el paciente ha realizado cada uno de los dos movimientos, puesto que depende del paciente y del grado de afectación del túnel carpiano va a ser capaz o no de realizar correctamente ambos movimientos.

El juego comienza cuando el paciente extiende la mano y con la aparición de forma aleatoria de una manzana roja (Figura 3.15.) o un cerdo (Figura 3.16.).

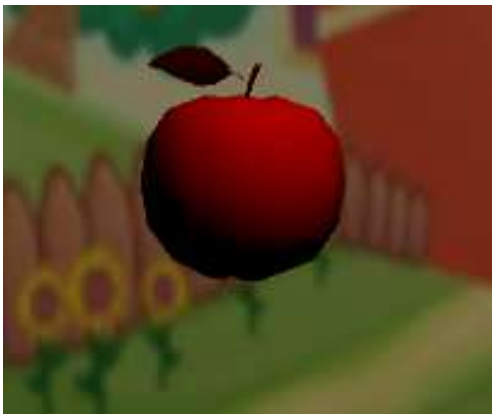
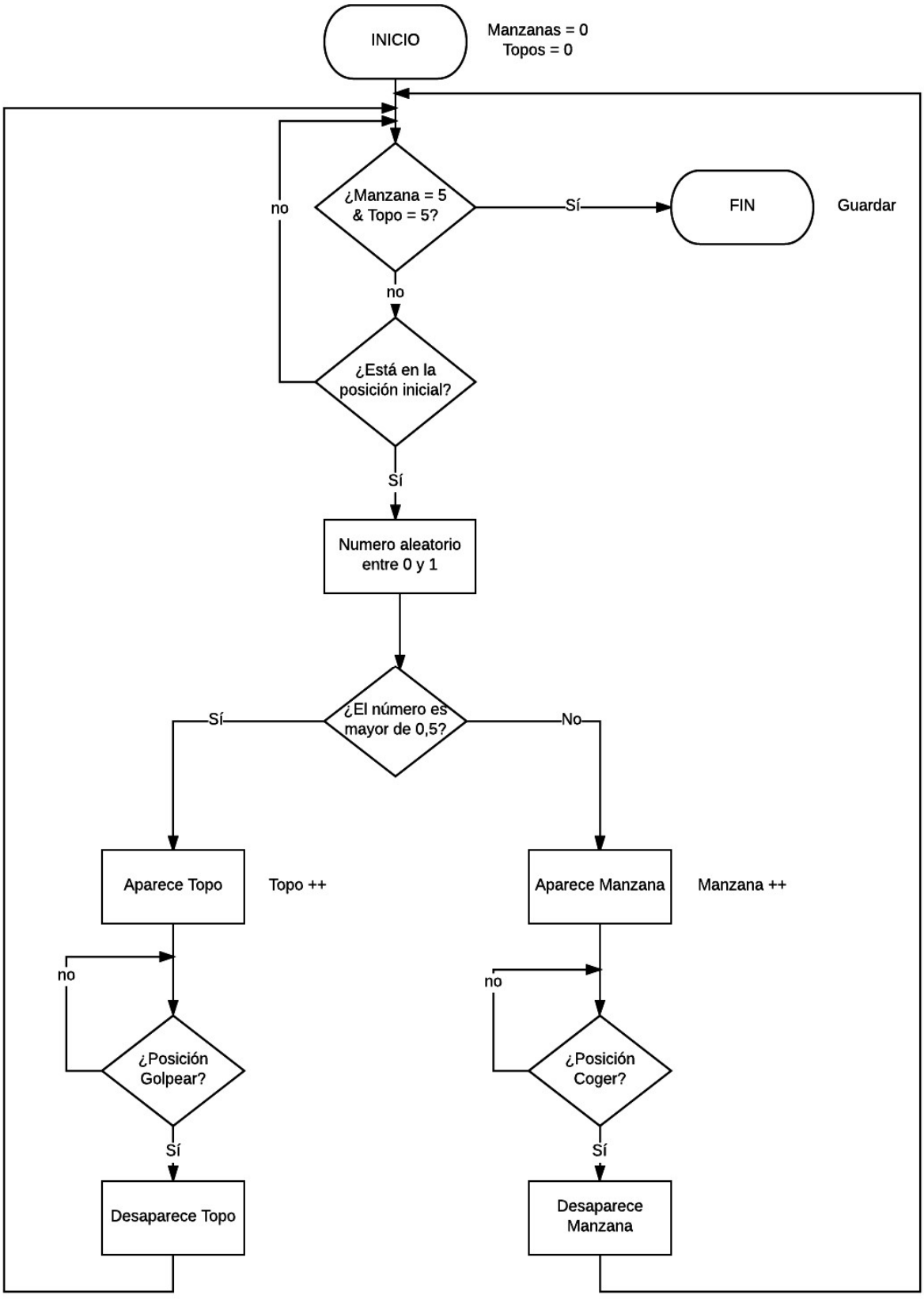


Figura 3. 15. Manzana Juego Granja

Figura 3. 16. Cerdo Juego Granja

En cuando aparece un cerdo el paciente tiene que realizar el movimiento de "Golpear" para dar al cerdo y que desaparezca. Una vez el paciente ha vuelto a la posición inicial, que es la de tener la mano extendida, aparecerá de forma aleatoria otro cerdo o

manzana. En el caso de aparecer una manzana el paciente tiene que hacer el gesto de "Coger" para que desaparezca la manzana y volver a la posición inicial para que aparezca otra pieza. Las dos figuras han sido elegidas por ser consideradas piezas neutras aptas para todos los públicos y fáciles de diferenciar. El juego termina cuando el paciente ha cogido 5 manzanas y ha golpeado a 5 cerdos. Una vez se ha terminado se crea un archivo donde se guarda el tiempo que se ha tardado en coger cada uno de los cerdos y de las manzanas para que los médicos puedan posteriormente analizar si se ha mejorado o no en la práctica de dicho juego. Todo esto se puede ver mejor en el siguiente flujograma.



Flujograma 3. 11. Juego Cazar Topo

Como se puede ver en el flujograma, el juego crea de forma aleatoria una manzana o un cerdo. En este caso como en la programación en vez de poner cerdo, pone topo por haberse considerado en un principio el animal a golpear, en el flujograma también pone topo. Cuando se Golpea o se Coge, dependiendo de cuál salga, desaparece la figura que había aparecido anteriormente. Al volver a la posición inicial se genera otra manzana o cerdo. Todo esto termina cuando ambos llegan a 5. Una vez uno de los dos ha llegado a 5 solo aparece la otra figura. Esto no se ha puesto dentro del flujograma para facilitar la comprensión del mismo.

También se ha añadido posteriormente un botón que permite que se dé por cogida o golpeada la figura que haya aparecido. Esto se ha añadido por si el paciente no fuera capaz de hacer el movimiento poder pasar al siguiente. En caso de pulsar el botón se añade un fallo, que también será cuantificado al acabar el juego como se puede ver de forma muy simple en el siguiente flujograma.

Flujograma 3. 12. Fallos Granja

3.4. INTERFAZ

Este proyecto está pensado para ser utilizado por médicos y fisioterapeutas que quieran evaluar el movimiento que pueden realizar los pacientes y también directamente por los pacientes. Por ello este *serious game* tiene que tener una interfaz sencilla que pueda ser entendida por todas las personas que lo vayan a utilizar. En este caso la interfaz sirve para unir las diferentes escenas de las que está compuesto el proyecto y a su vez para permitir al que interactúa con el juego modificar ciertos aspectos del mismo para conseguir una aclimatación mayor.

La interfaz sirve para editar los datos de cada paciente. También es la forma de unir todas las partes del *serious game* explicado anteriormente. Por ello dentro del programa Unity en el archivo Menú, que es donde están todas las escenas unidas podemos encontrar las siguientes escenas (ver figura 3.17.).

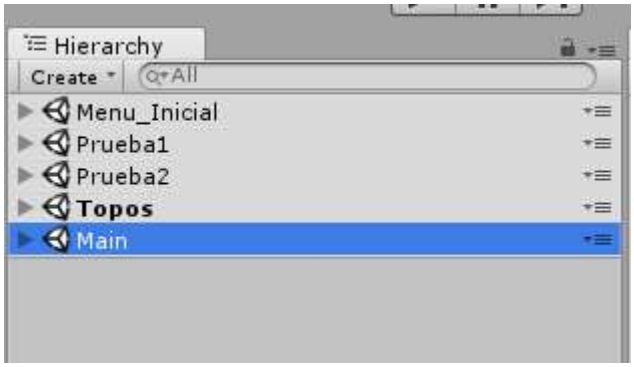


Figura 3. 17. Escenas del *Serious Game*

- Menu_Inicial: esta escena sirve para unir todas las anteriores y en ella se guardan los diferentes datos del paciente. Será explicada una vez queden nombradas todas las escenas.

- Prueba1: este es el nombre que tiene la prueba de evaluación de las manos. En este caso y para conseguir una mayor complicidad por parte del paciente se ha añadido una barra horizontal cuando se realiza el movimiento horizontal y una vertical cuando se realiza el movimiento vertical que se desplazará como se desplace la mano. Esto se puede ver en la figura 3.18. También se han añadido

botones y cambiado el aspecto gráfico para facilitar el entendimiento del juego, como en el resto de juegos.

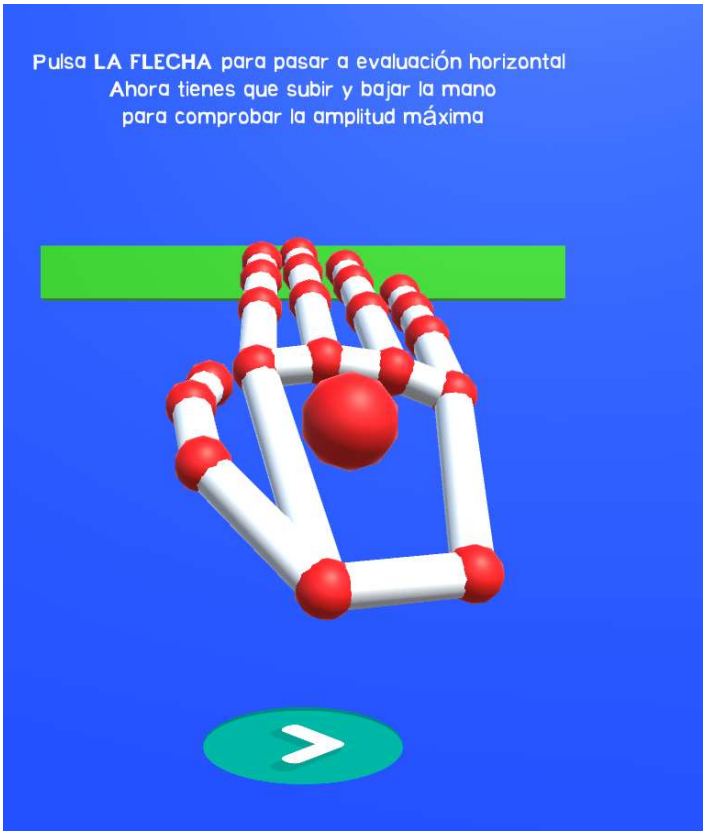


Figura 3. 18. Mano con Barra

- Prueba2: es la evaluación de los dedos. En este caso se ha modificado levemente los scripts para en vez de pedir el ángulo máximo de las falanges proximales y medias se pida que se ponga la mano en las posiciones que mejor vienen para la rehabilitación del síndrome del túnel carpiano, que son las que anteriormente se han llamado Coger y Golpear.

- Topos: este es el *serious game* de la granja explicado anteriormente una vez unido al resto de escenas. En este caso se puede elegir el número de cerdos y de manzanas que aparecen, siendo un número variable entre 0 y 10.

- Main: por último, hay que explicar el main o juego principal. Este es el Space Shooter 2.0. Es llamado main por ser en un principio considerado el juego

principal sobre el que girase el proyecto, mas luego ha sido modificado siendo solo una parte del mismo.

Todas estas escenas anteriormente tenían un apartado en el que se preguntaba el nombre del paciente y solo se podían hacer con la mano derecha. Todo esto ha sido modificado para obtener la información del Menu_Inicial, que es donde se dejarán claros los datos del paciente.

3.4.1. EXPLICACIONES

Para poder jugar a cada uno de los apartados de los que se compone el *serious game* es necesario conocer como funcionan. Si esta rehabilitación se lleva a cabo dentro del centro de rehabilitación o con una persona que comprenda como funciona, no es necesario ninguna explicación. Mas como este juego está pensado para que pueda ser utilizado por los pacientes en casa, y teniendo en cuenta que el público al que está dirigido en este caso es muy amplio, puede darse el caso de personas que tras haber realizado los juegos varias veces no se acuerden de cómo funcionaba el juego y por eso se ha incluido una explicación antes de empezar cada escena, o entre medias de ellas para facilitar la comprensión por parte de todo el mundo del juego. Un ejemplo de ello se puede ver en la explicación anterior de Space Shooter 2.0 y otro ejemplo en el juego de la granja es el que se puede ver en la figura 3.19.

Figura 3. 19. Juego Granja con explicación

3.4.2. MENU_INICIAL

Esta escena consta de varios submenús en los que poder modificar o crear datos del paciente, así como donde elegir el juego que se quiere realizar, que son los que se pueden ver en la figura 3.20.

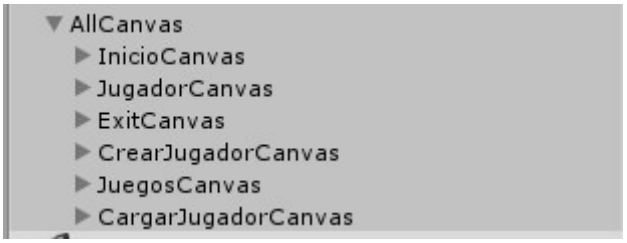


Figura 3. 20. Canvas del Menu_Inicial

- Exit: en todos los subapartados en la parte superior derecha encontraremos el botón Exit con el que poder salir del juego.

- Inicio: Al arrancar el juego entramos en el apartado Inicio en el cual aparece un botón con el símbolo de Play con el que dar comienzo al juego y otro botón con el doble triángulo para hacer todas las fases del juego seguidas, como se puede ver en la figura 3.21.

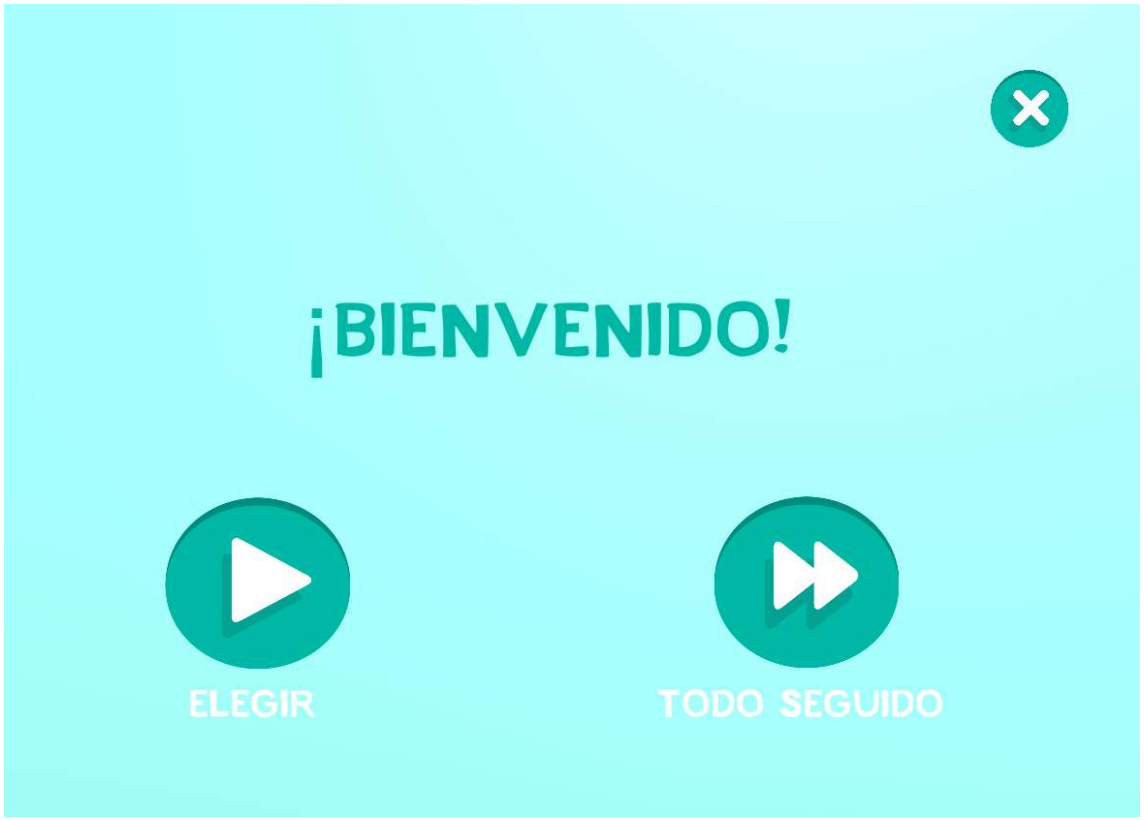


Figura 3. 21. Menu_Inicial Inicio

- Jugador: iniciado el programa entramos en una sección en la que decidimos si crear un nuevo jugador (Soy Nuevo) o cargar uno ya creado (Tengo Cuenta). En función del botón que elijamos iremos a un menú o a otro. Ver figura 3.22.

Figura 3. 22. Menu_Inicial Jugador

- Soy Nuevo: si se ha pulsado este botón se llega a una zona en la cual te pide el nombre y apellidos del paciente, el motivo de la rehabilitación y la mano lesionada. Una vez rellenados los datos, y si la mano es derecha o izquierda, se procede a guardar o cancelar lo escrito como se puede ver en la figura 3.23. Si se pulsa Play, que es el botón que guarda, se llega al menú Juegos. En caso de pulsar la cruz se vuelve al anterior menú.

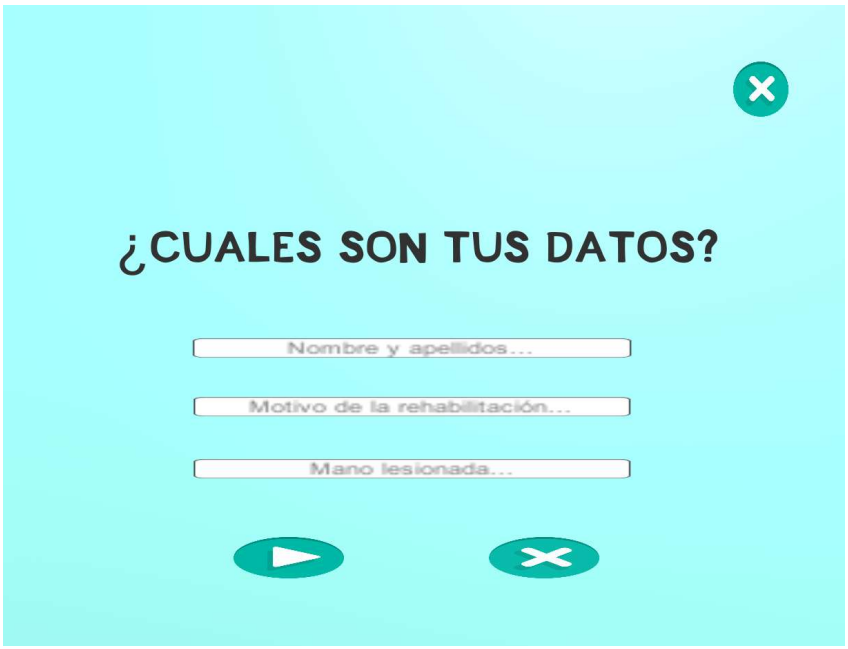


Figura 3. 23. Menu_Inicial CrearJugador

- Tengo Cuenta: si se quiere realizar una evaluación o juego con un paciente que ya exista se pulsa el botón de cargar jugador. En esta opción se procede a escribir el nombre del paciente y comprobar si existe. En caso de que exista se pasa al menú Juegos. Ver figura 3.24.

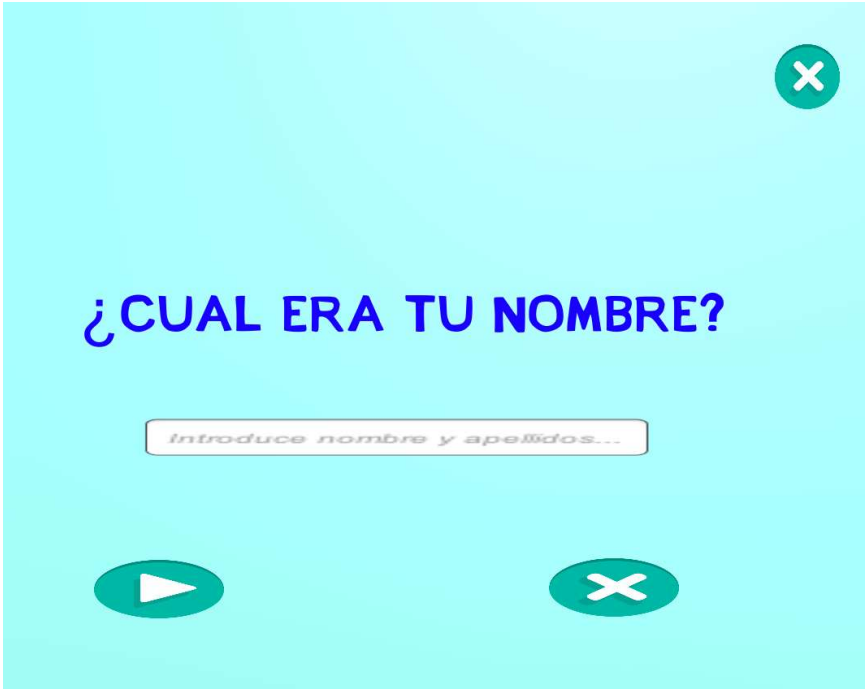


Figura 3. 24. Menu_Inicial CargarJugador

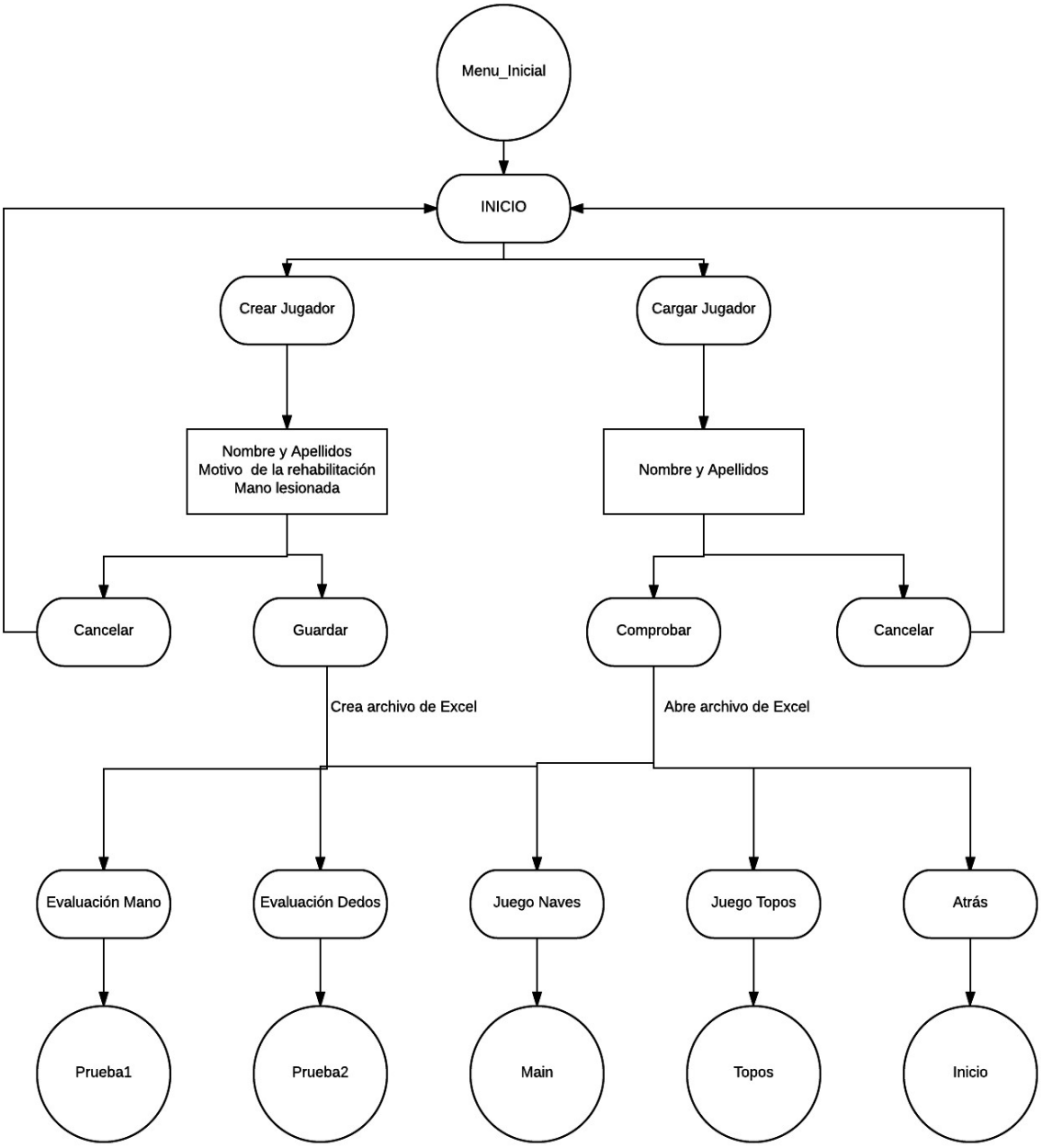
- Juegos: una vez se ha llegado aquí aparecen varios botones, para poder elegir entre los distintos juegos o evaluaciones que se han hablado anteriormente. Los botones son los que se pueden ver en la figura 3.25. Siendo Juego Naves el Space Shooter 2.0 y Juego Granja el Cazar Topo.

Figura 3. 25. Menu_Inicial Juegos

El Menu_Inicial sirve a su vez para saber de qué mano está lesionado el paciente y así solo permitir la interacción en cada juego por parte de la mano lesionada.

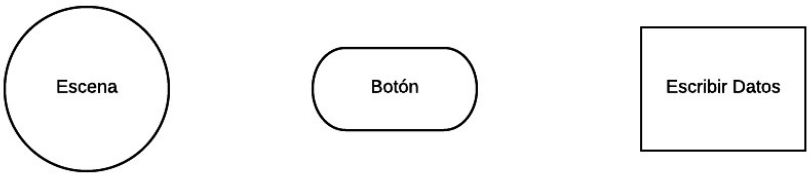
Anteriormente se ha explicado que la abducción era el ángulo negativo vertical y la aducción era el positivo. Esto era dentro del contexto anterior en el cual solo se contemplaba el uso de la mano derecha. Una vez se puede escoger la mano el ángulo positivo será siempre el que haga girar la mano a la derecha, independientemente de si es el movimiento de abducción o de aducción.

En el siguiente diagrama de flujo se puede ver cómo funciona el Menú Inicial:



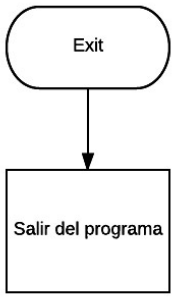
Flujograma 3. 13. Menu_Inicial

En este caso no es un flujograma al uso, si no que se trata de un diagrama el cual cada forma representa lo siguiente:



Flujograma 3. 14. Representación Formas

El botón de *Exit* está presente en todo momento como se puede ver en el siguiente diagrama.



Flujograma 3. 15. Botón Exit

Se ha escogido realizar este tipo de diagrama para una mayor comprensión y claridad del Menú_Inicial.

Todos los juegos y evaluaciones dejan una serie de métricas cuantificables que permiten a los médicos tener una referencia y poder actuar en consecuencia a esos datos. Cada vez que se crea un paciente se crea un archivo de Excel donde se guardan todos los datos.

3.4.3. GUARDADO

Todas las escenas del *serious game* dejan una serie de datos que son interesantes conocer por parte de los médicos y por ello se crea un archivo con cada paciente en el que se guardan los datos que se pueden ver en el ejemplo de la figura 3.26.

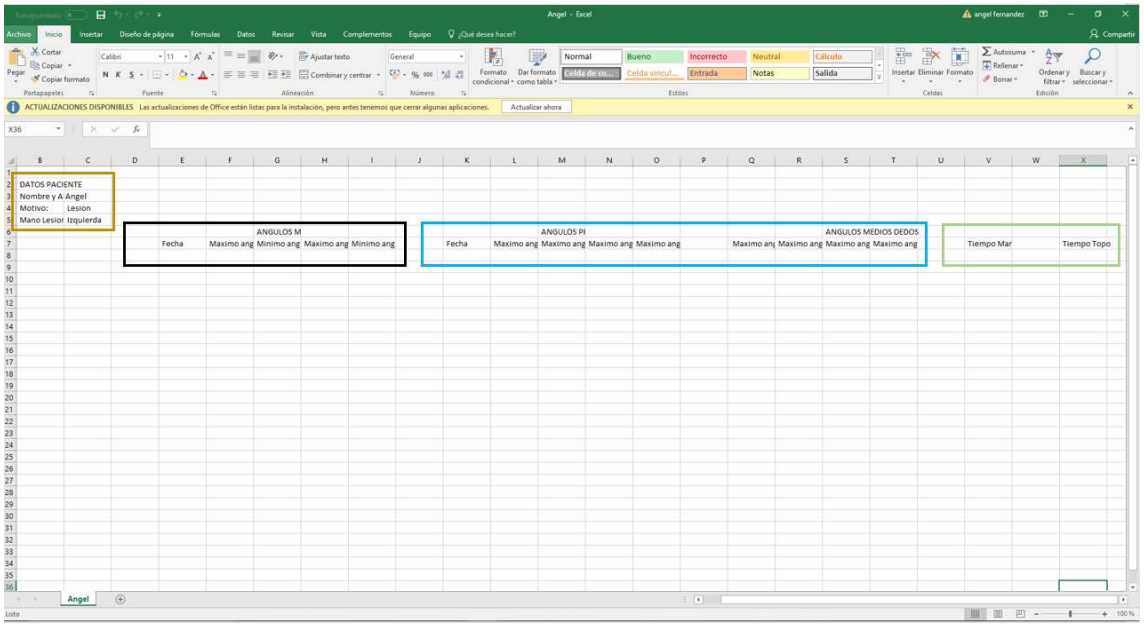


Figura 3. 26. Archivo Excel

Para mayor claridad se enseñará de forma ampliada donde se guardan cada uno de los datos.

- Datos paciente: en el recuadro en naranja van los datos del nombre y apellidos del paciente, así como el motivo de la lesión y la mano lesionada como se puede ver en la figura 3.27.

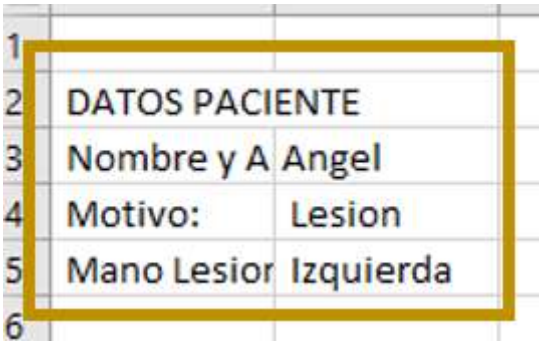


Figura 3. 27. Datos paciente

- Ángulos Mano: aquí, en el recuadro negro, se guardan la fecha en la que se ha realizado la evaluación de la mano y los datos de los ángulos obtenidos. En caso de realizarse varias veces la evaluación se guarda en la siguiente línea los datos de la segunda sesión.

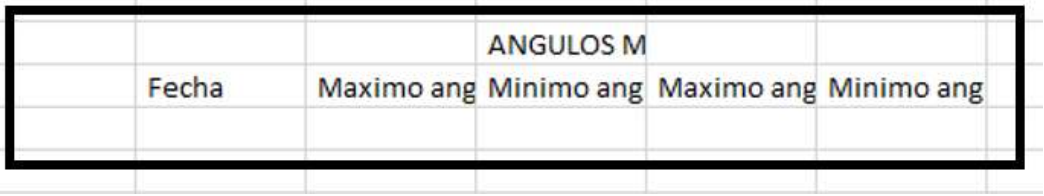


Figura 3. 28. Ángulos Mano

- Ángulos Dedos: en el recuadro en azul se guarda la fecha y los ángulos obtenidos de las distintas falanges en la evaluación dedos, como se puede ver en la figura 3.29.

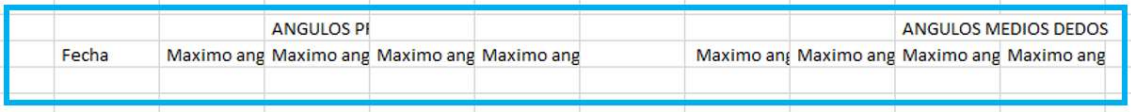


Figura 3. 29. Ángulos Dedos

- Tiempo Topos: en el último recuadro, que se ha señalado en verde, se guarda el tiempo en milisegundos que se tarda desde que aparece cada topo o manzana hasta que se Coge o se Golpea el mismo. También se guardan los fallos que se tiene.

Figura 3. 30. Tiempo Topos

4. RESULTADOS

Los resultados de la realización de este proyecto dependen, en cierta medida, de la utilidad que tenga dicho *serious game*. Esto en un principio no se puede saber hasta que no se haga un estudio con pacientes para obtener un *feedback* por su parte y poder así modificar o hacer consideraciones. Aun así la evaluación y los juegos dependen del cálculo de los ángulos obtenidos. Por ello hay que saber cómo se han calculado los ángulos y la precisión de los mismos para hacer el juego acorde.

4.1. CÁLCULO ÁNGULOS

A la hora de cuantificar el ángulo que tiene la mano se han tenido que realizar una serie de cálculos tanto para saber los movimientos de la muñeca como de los dedos.

4.1.1. ÁNGULOS MANO

La información necesaria se ha sacado de la Api del *Leap Motion* [21]. Dentro de ella se encuentra la clase Hand que nos permite obtener el vector al que apunta la palma de la mano y su vector perpendicular como se puede ver en la figura 4.1.

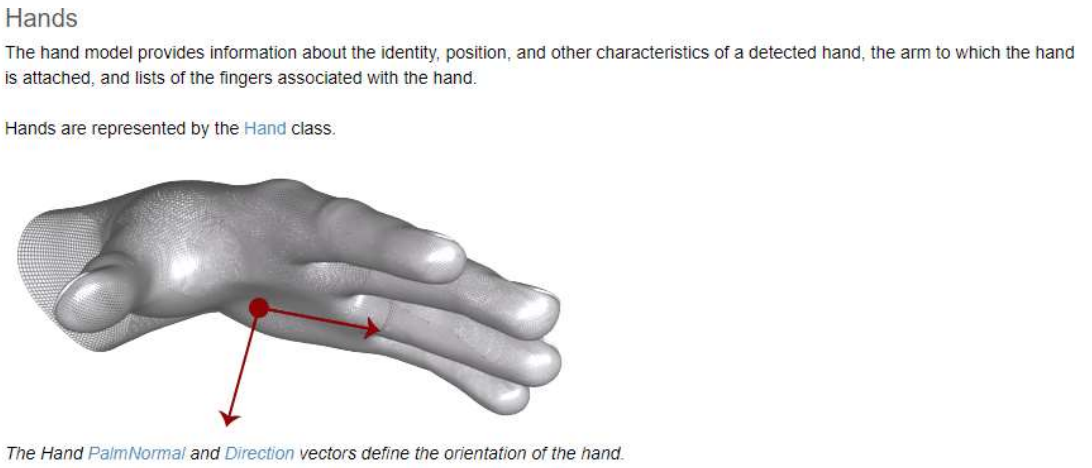


Figura 4. 1. Vectores Hand

Gracias a ello podemos mediante cierta lógica obtener el ángulo que tiene la mano respecto al suelo (o mesa o donde esté apoyado el *Leap Motion*) para saber el ángulo que representa la normal y así saber cuánto sube o baja la mano respecto de la horizontal.

Esto se consigue mediante la lógica implementada como se puede ver en la figura 4.2.

```
palmVector = hand.PalmNormal.ToVector3();
palmVector2 = hand.Direction.ToVector3();
horizonte.Set(1, 0);
palma.Set(palmVector.z, palmVector.y);
angle = (Vector2.Angle(horizonte, palma) - 90) * -1;
angle2 = palmVector2.x * 90;
```

Figura 4. 2. Lógica cálculo ángulos manos

Aquí se puede ver como *palmVector* es el conjunto de 3 vectores que representa la normal de la mano. En él si queremos saber el ángulo tenemos que saber como está orientada nuestra mano en el *Leap motion*. Sacando una captura del eje de coordenadas obtenemos lo siguiente:

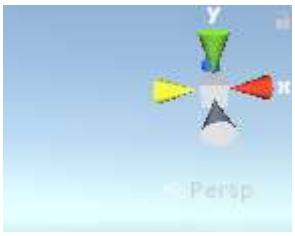


Figura 4. 3. Coordenadas Unity

En nuestro caso y teniendo en cuenta donde está la cámara nos quedan las coordenadas tal que así:

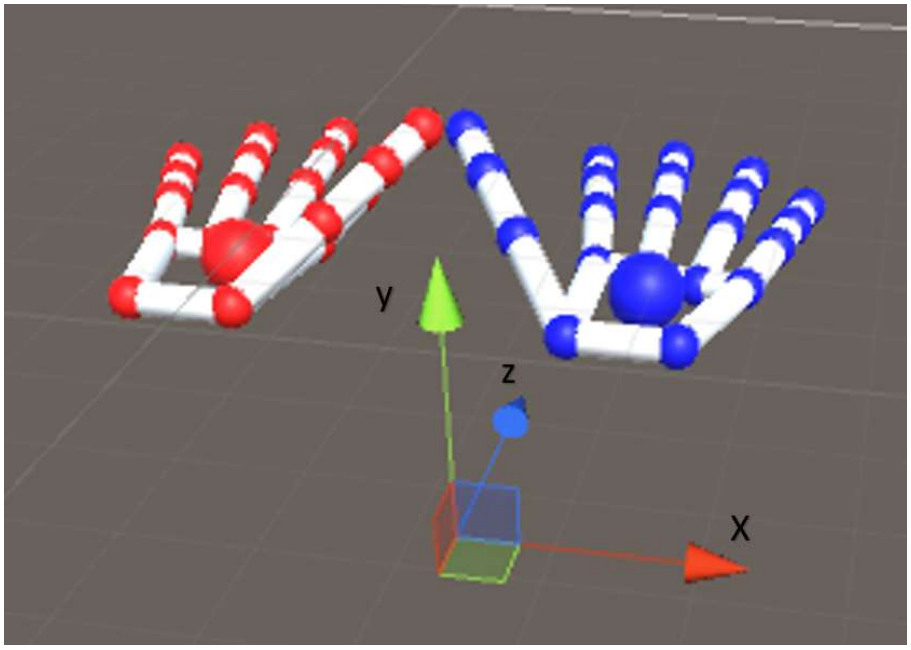


Figura 4. 4. Coordenadas Unity con las manos

Una vez tenemos claras las coordenadas y sabiendo que lo que queremos obtener para saber cuánto sube o cuánto baja la mano es el ángulo que forma el eje Z con el Y de la figura 4.5.

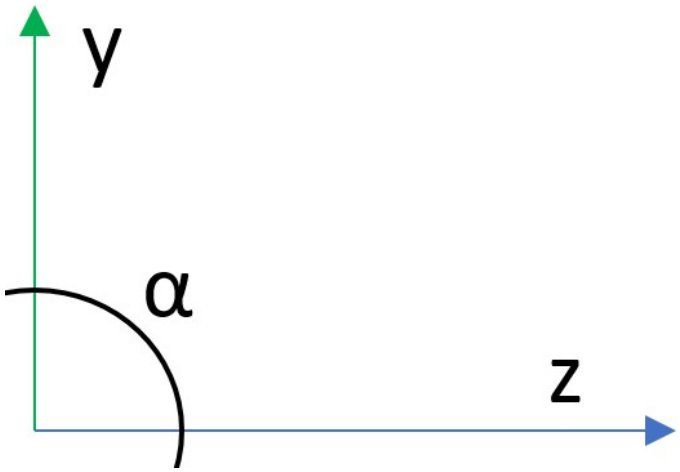


Figura 4. 5. Coordenadas calculo ángulo manos

Para obtener el ángulo que forma la palma con el eje Z se han usado las fórmulas vistas en la figura 4.2. En ellas el conjunto de 3 vectores palmVector corresponde a la normal de la mano.

Este vector tiene la misma dirección pero distinto sentido que el eje Y cuando la palma está mirando hacia abajo como se puede observar en la siguiente figura:

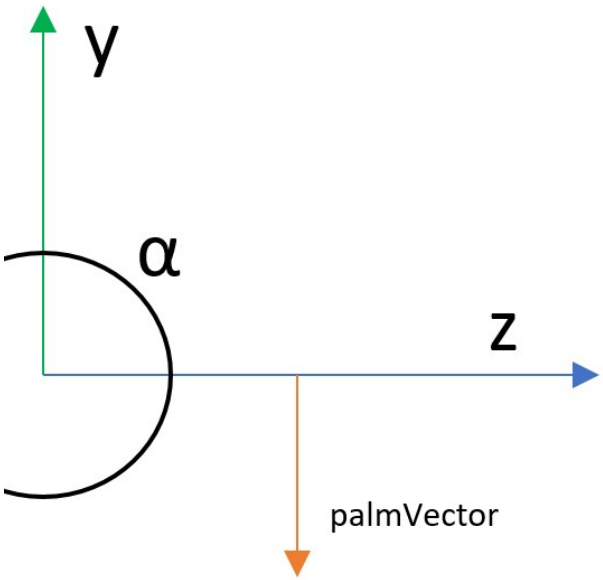


Figura 4. 6. palmVector

En él se puede ver que el ángulo que el ángulo que forma PalmVector con el eje Z.

Dentro de estas líneas de código se ha implementado un Vector2 llamado horizonte que corresponde al eje Z de Unity para comparar ese vector con el que obtenemos de la normal de la mano y así obtener el ángulo.

```
horizonte.Set(1, 0);
```

Posteriormente se ha pasado los ejes Z e Y de nuestra mano en Unity a los ejes normales que serían el eje X e Y gracias a la función Set que se ha utilizado en un Vector2 que se ha denominado Palma como se puede ver en la siguiente línea de código:

```
palma.Set(palmVector.z, palmVector.y);
```

Tras esto se calcula el ángulo que forman ambos vectores y con ello se sabe el ángulo que forma la mano, que en nuestro caso es angle y sirve para calcular Vertical Min y Vertical Max.

Para el cálculo del ángulo que tiene la mano cuando se realiza el giro de abducción o de aducción se ha utilizado la misma lógica pero con la conversión necesaria para representar los vectores en las coordenadas que estos tienen en Unity, que son las que se pueden ver en la figura 4.7.

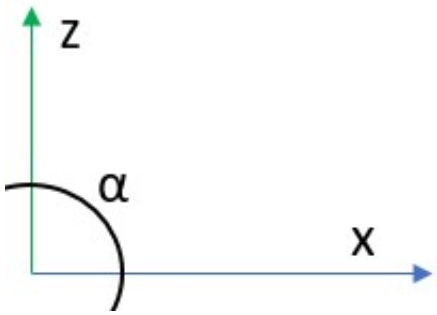


Figura 4. 7. Coordenadas Giro derecha e izquierda

Con lo cual el código resultante es el siguiente (figura 4.8.):

```
palmVector = hand.PalmNormal.ToVector3();
palmVector2 = hand.Direction.ToVector3();
horizonte.Set(1, 0);
palma.Set(palmVector.z, palmVector.y);
angle = (Vector2.Angle(horizonte, palma) - 90) * -1;
horizonte2.Set(1, 0);
palma2.Set(palmVector2.x, palmVector2.z);
angle2= (Vector2.Angle(palma2, horizonte2)-90);
```

Figura 4. 8. Código ángulo mano

4.1.2. ÁNGULOS DEDOS

Los ángulos calculados en este caso son los que forman las dos primeras falanges, posteriores a los metacarpianos, de cada dedo de la mano [22]. A excepción del dedo pulgar, que no movimiento, si bien es muy influyente en el síndrome del túnel carpiano, no se ha tenido en cuenta a la hora de hacer el juego de los topos ni la evaluación de los dedos por falta de precisión y por no girar en la misma dirección que el resto de los dedos. Los vectores de las falanges utilizadas son las que se pueden ver en la siguiente figura:

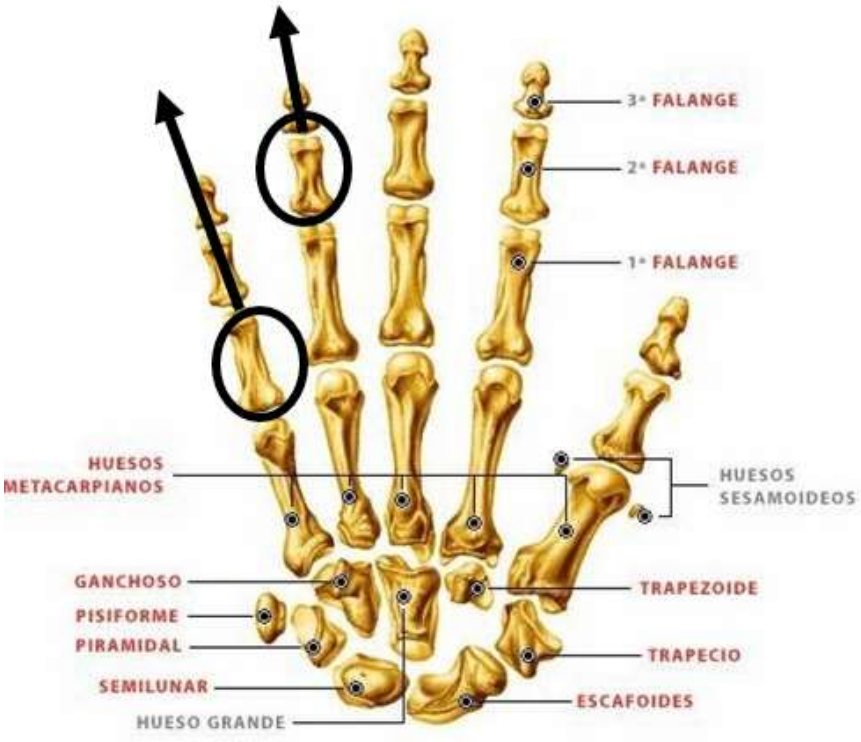


Figura 4. 9. Huesos Mano

En esta figura se ha rodeado la primera falange del dedo meñique y su correspondiente dirección, que es la que se quiere calcular y la segunda falange del dedo anular.

Para poder medir dicha dirección utilizando el *Leap Motion* es necesario saber de qué clases dispone la mano dentro de dicho aparato. Ahondando dentro de la API del *Leap Motion* para Unity nos encontramos con la clase *Finger* como se puede ver en la figura 4.10.

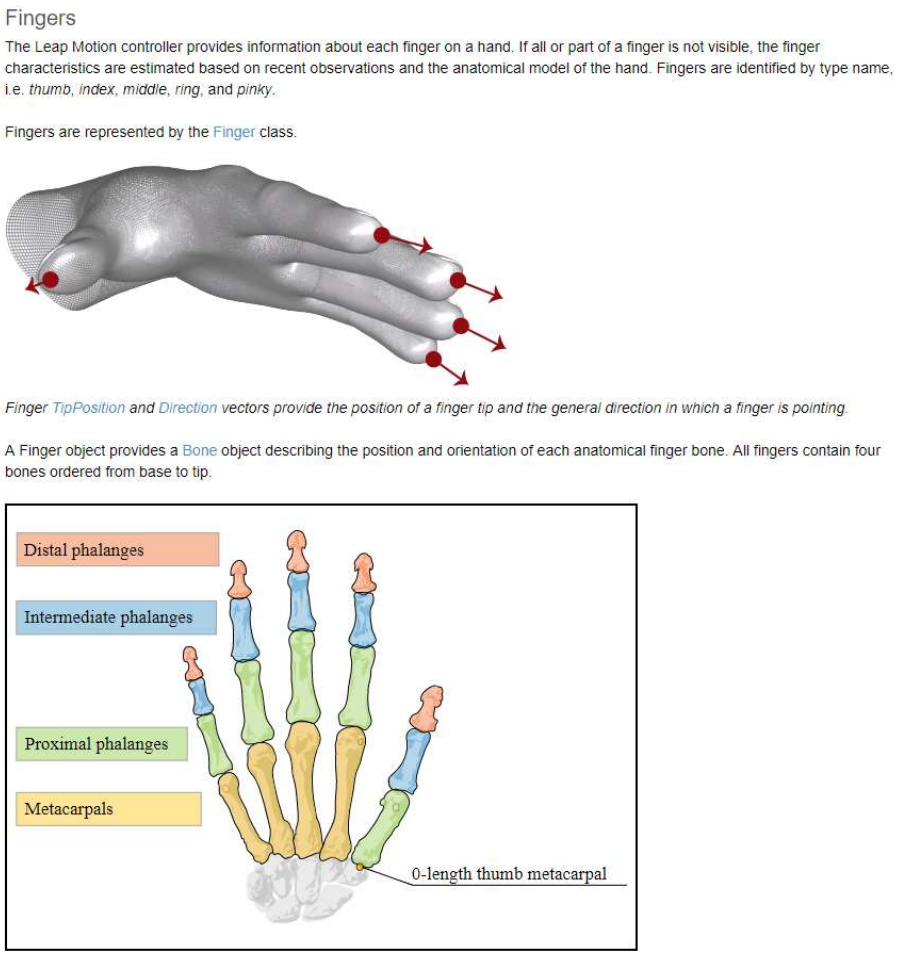


Figura 4. 10. Vector dedos Unity

El cálculo del ángulo que tiene cada falange se ha realizado de igual forma que con los de la palma de la mano, pero en este caso solo para el movimiento de subir y bajar los dedos. Esto es posible al tener la clase Fingers también la posibilidad de saber en qué dirección apuntan con un vector.

En este caso el sistema de coordenadas es el de la siguiente figura.

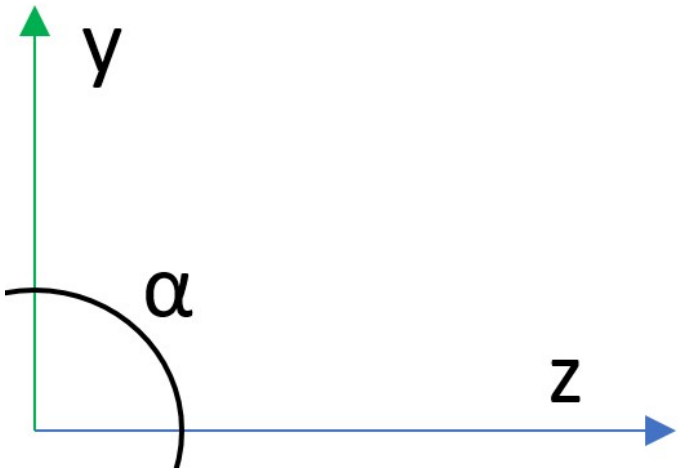


Figura 4. 11. Coordenadas Dedos Mano

4.2. PRECISIÓN ÁNGULOS MANO

Una de las ventajas que tiene este sistema de evaluación respecto al convencional es que se puede medir el ángulo que tiene la mano. Para que este sistema represente una ventaja y pueda ser utilizado es necesario que los ángulos obtenidos sean precisos. Por ello se han realizado una serie de pruebas con distintas variables para comprobar la fiabilidad de los mismos.

4.2.1. DISTANCIA MANO

Para comprobar cuál es la distancia óptima a la que poner el *Leap Motion* se han realizado el movimiento de subir y bajar la mano con el antebrazo quieto y desplazar la misma hacia derecha e izquierda. En todas las pruebas se ha usado la mano derecha como

mano de ejemplo por considerar que los mismos datos y errores se van a obtener en ambas manos.

4.2.1.1. Subir y bajar la mano

El experimento ha consistido en subir y bajar la mano a distintas alturas y ver si la respuesta obtenida en forma del ángulo que nos da la palma de la mano es la esperada con cada distancia. La respuesta que se presupone es la de una señal parecida a la senoidal en la que el ángulo baje de 0º siendo negativo, subiendo posteriormente y siendo positivo. Esta señal es la esperada porque le movimiento hecho es el de bajar y subir la mano. Se ha medido la respuesta a las siguientes distancias: menos de 10 cm, 20 cm, 25 cm y 30 cm.

Gráfico 4. 1. Subida y Bajada a menos de 10 cm

La primera gráfica corresponde a realizar el movimiento a menos de 10 cm de distancia entre el *Leap Motion* y la mano en altura. En este caso se puede ver como la señal tiene picos y resultados no esperados prácticamente en la mitad del dominio. Esto se debe al hecho de que, a esa distancia, una vez se mueve la mano, sale del radio de acción del dispositivo y por tanto el programa no sabe dónde está la mano.

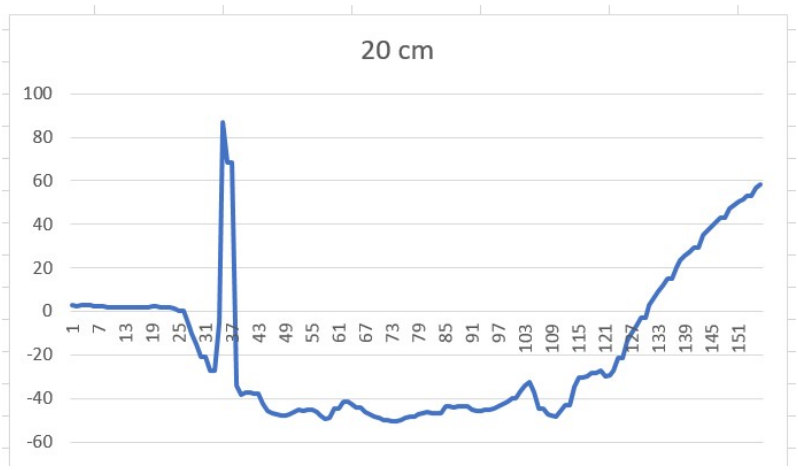


Gráfico 4. 2. Subida y Bajada a 20 cm

En este caso se puede observar que ya no hay tantos resultados dispares. Al estar a una distancia mayor el *Leap Motion* es capaz de detectar la mano en casi todo momento, si bien hay un pico en el cual pierde la referencia.

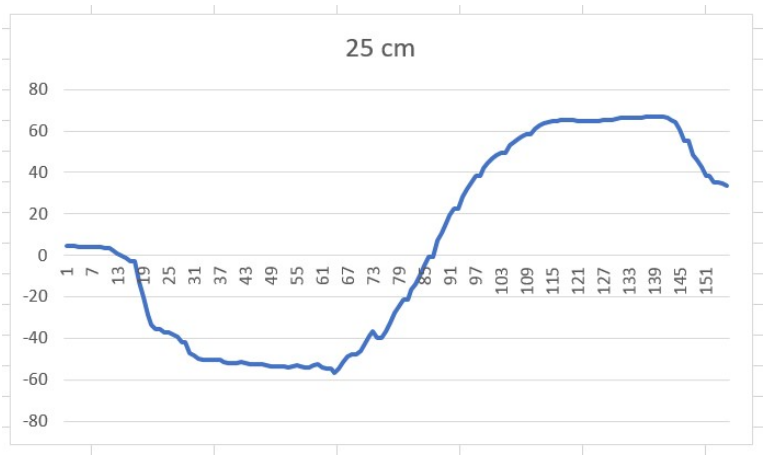


Gráfico 4. 3. Subida y Bajada a 25 cm

En la distancia de 25 cm se puede ver una señal muy parecida a la senoidal. En este caso no existe ningún pico. La gráfica tiene cierto ruido puesto que al estar captando el valor del ángulo cada poco tiempo no es posible ser del todo preciso, y también por ser un movimiento hecho por una persona, que también puede añadir error humano a dicho movimiento.

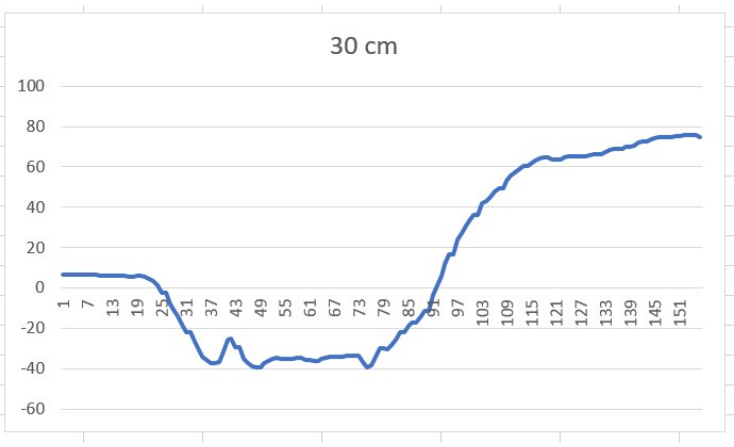


Gráfico 4. 4. Subida y Bajada a 30 cm

Por último, esta gráfica, la de 30 cm, no tiene gran diferencia respecto a la anterior aunque con un leve pico.

Al comparar todas las gráficas se puede ver como la distancia más óptima es la de 25 cm. En las gráficas a menor distancia existen muchas ocasiones en las que la gráfica tiene altibajos y picos que no deberían dar. A medida que la distancia se aproxima a la de 25 cm la gráfica va pareciéndose más a una gráfica senoidal. A su vez es mejor la distancia de 25 cm que la de 30 cm por el hecho de que a mayor distancia entre la mano y el dispositivo más difícil es encontrar un soporte o posición cómoda en la que colocarlo.

4.2.1.2. Girar mano

Un segundo experimento llevado a cabo con el que comprobar que distancia es la óptima para colocar la mano ha consistido en volver a tomar datos como en el anterior caso, pero en este caso al realizar con la mano los movimientos de abducción y de aducción. En este caso, al comprobar que las gráficas tienen todas cierto ruido, se ha optado por comprobar el valor que se obtiene en las siguientes distancias: menos de 10 cm, 15 cm, 20 cm, 25 cm y 30 cm.

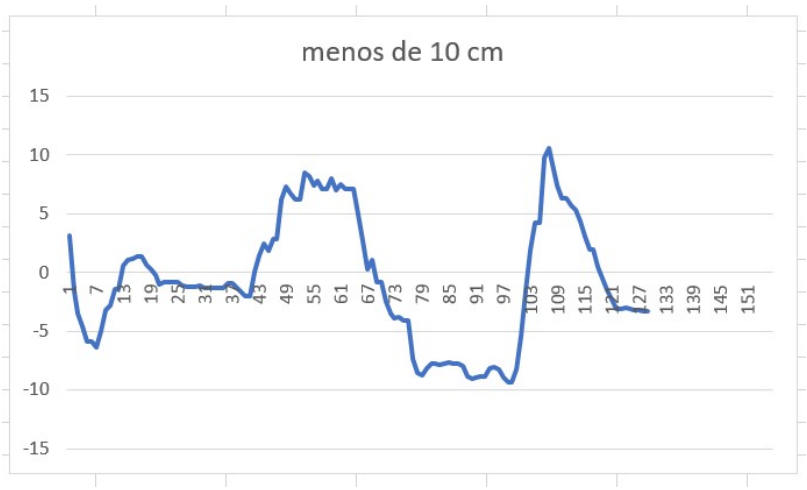


Gráfico 4. 5. Derecha e Izquierda a menos de 10 cm

En esta primera gráfica, a menos de 10 cm se puede apreciar como los cambios que hay no son graduales y aunque la señal presenta una forma parecida a la senoidal, no es precisa y tiene mucho ruido.

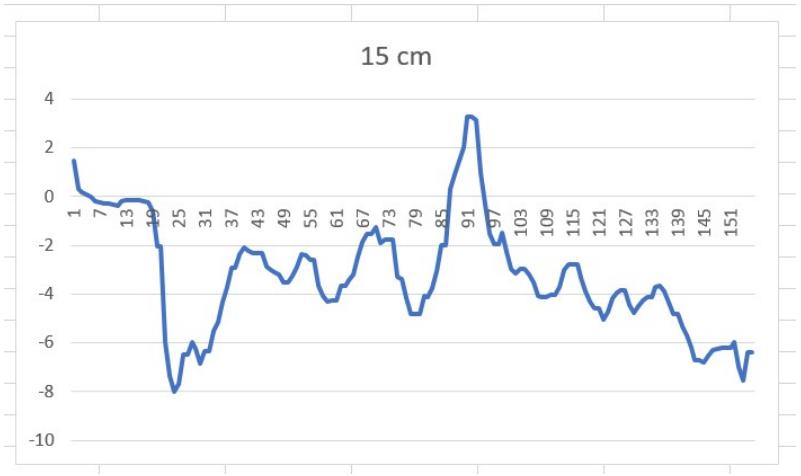


Gráfico 4. 6. Derecha e Izquierda a 15 cm

En la gráfica X se adquiere un valor positivo para el ángulo en una sola ocasión más el estado inicial. Esto hace que no se pueda considerar como correctos los resultados obtenidos por la gran cantidad de datos negativos obtenidos.

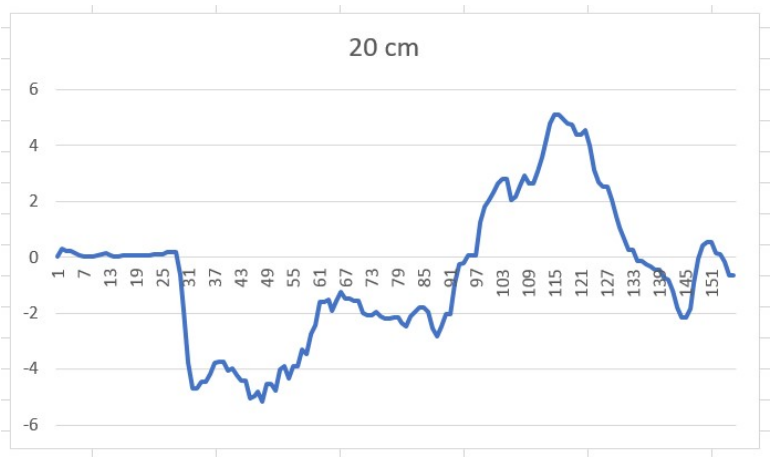


Gráfico 4. 7. Derecha e Izquierda a 20 cm

En la distancia de 20 cm existe ruido y la respuesta obtenida no sigue la forma de una señal senoidal.

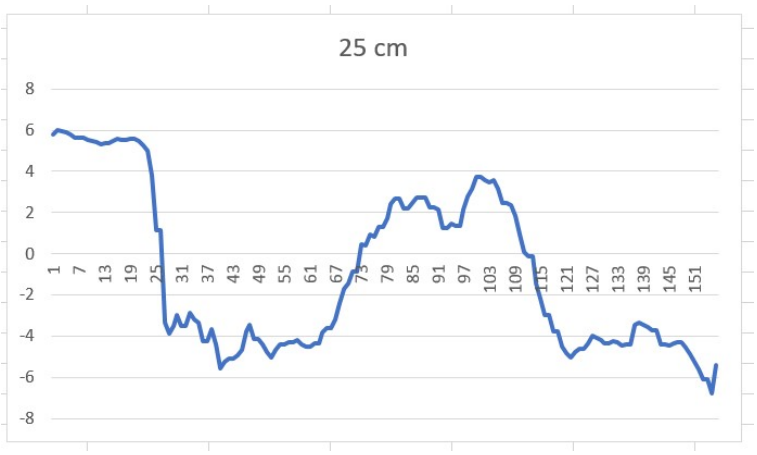


Gráfico 4. 8. Derecha e Izquierda a 25 cm

En la gráfica 4.8. se puede ver como adquiere la forma adecuada, si bien tiene bastante ruido y la posición inicial tiene un alto valor en positivo, esto es debido a iniciar el movimiento sin tener la mano como continuación del eje axial del antebrazo.

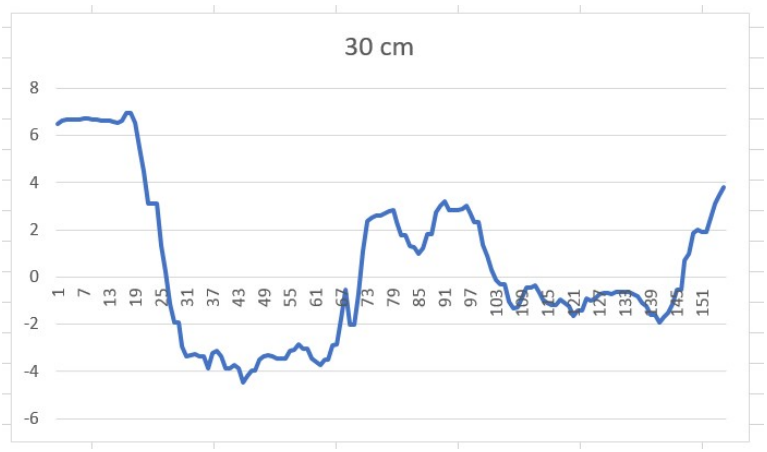


Gráfico 4. 9. Derecha e Izquierda a 30 cm

En esta última gráfica se puede apreciar una forma parecida a la anterior. Esto es debido a que en ambos casos se detecta en todo momento la mano por estar dentro de la zona de detección del *Leap Motion*.

Gracias a haber realizar anteriormente otra prueba con otro movimiento y con la premisa de que a menor distancia del *Leap* mejor para la comodidad y ergonomía del posible soporte se ha llegado a la conclusión de que la mejor distancia es la de 25 cm. Como no se va a hacer un cambio de distancia a mitad de la prueba había que llegar a una misma conclusión en ambas pruebas, que en este caso ha resultado ser la distancia de 25 cm.

4.2.2. SOPORTE

Dentro del juego hay que tener en cuenta la posibilidad de hacer un soporte puesto que tener el brazo en todo momento en el aire puede resultar agotador. También hay que considerarlo para añadir precisión al ejercicio, puesto que al mantener fijo el antebrazo conseguimos que no existan fallos por moverlo sin querer.

Para comprobar si se tienen más fiabilidad los datos con o sin soporte se ha utilizado un soporte improvisado que mantiene el antebrazo quiero, dejando la mano al descubierto para que la detecte el dispositivo y a una distancia cercana a la considerada mejor en el anterior experimento.

Primero, se ha graficado el movimiento de subir y bajar la mano.

Gráfico 4. 10. Subir y Bajar a 25 cm con apoyo

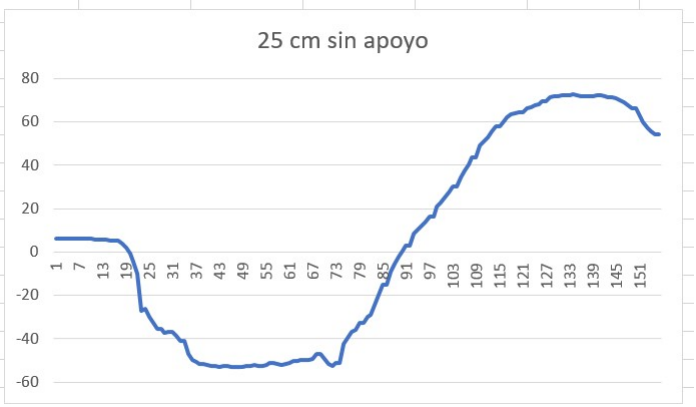


Gráfico 4. 11. Subir y Bajar a 25 cm sin apoyo

En ambas gráficas se puede ver una figura parecida. Si bien, se puede notar como los valores máximos y mínimos obtenidos no son iguales teniendo una diferencia de aproximadamente 10º tanto en subida como en bajada. Para comprobar cuál de los dos es más preciso se ha procedido a capturar en 5 ocasiones distintas el valor del ángulo máximo obtenido al hacer el movimiento de extensión con y sin apoyo.

4.2.2.1. Ángulo máximo de extensión

Tomando 5 muestras del valor máximo de extensión de un sujeto con soporte obtenemos los siguientes resultados.

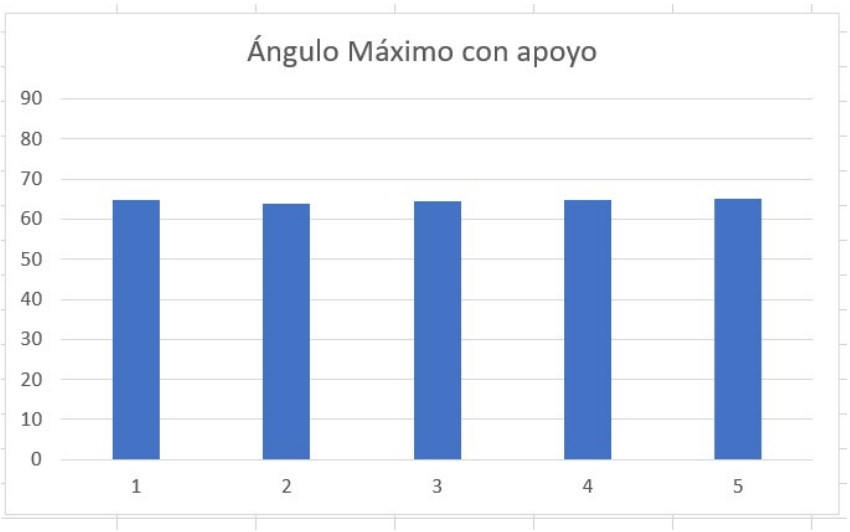


Gráfico 4. 12. Ángulo Máximo con apoyo extensión

Los resultados de forma numérica son:

1- 64, 68034º
2- 63, 81633º
3- 64, 41325º
4- 64, 8847º
5- 65, 09875º

El ángulo que se ha obtenido es bastante parejo en todas las ocasiones dando un porcentaje de diferencia máximo entre el mayor y mínimo valor del 2%.

Al ser comparado con las 5 muestras del valor máximo de extensión de un sujeto sin soporte obtenemos los siguientes resultados.

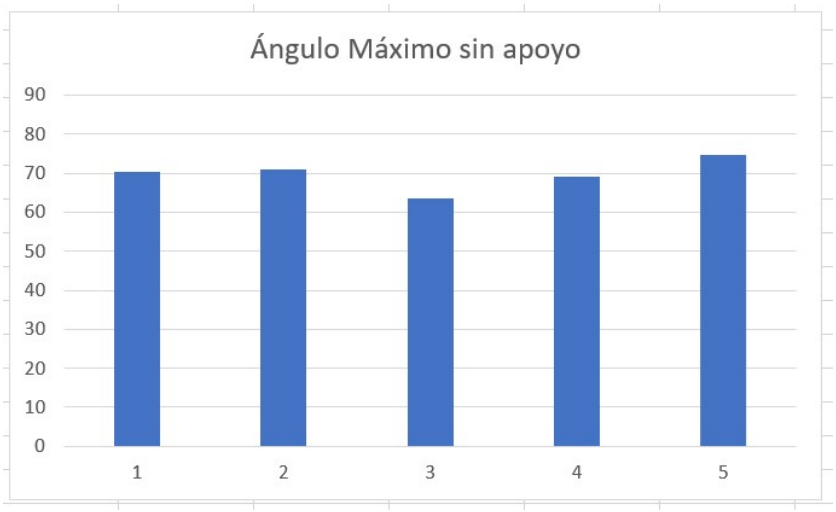


Gráfico 4. 13. Ángulo Máximo sin apoyo extensión

Los resultados de forma numérica son:

1- 70, 50348º
2- 70, 9036º
3- 63, 46013º
4- 69, 09755º
5- 74, 76762º

En este segundo caso se puede notar que hay mayor disparidad en los resultados obtenidos, dando por correcta la teoría de que el soporte influye de forma notable a la hora de medir los ángulos por mantener fijo el antebrazo. De forma cuantitativa la diferencia entre el máximo valor y el mínimo es del 17,8%.

Este valor obtenido es mucho mayor que con apoyo y por tanto se llega a la conclusión de que es necesario utilizar un soporte para obtener más precisión. A su vez al haber sido realizado este experimento con un paciente sano, que no tiene una excesiva fatiga, demuestra que la diferencia es más que notable. Si se hace este mismo experimento en personas con el tendón dañado seguramente la diferencia fuera mayor aún y por tanto concluimos con la necesidad de mantener el antebrazo fijo.

Para conseguir eso, se ha hecho uso de un soporte realizado anteriormente en el laboratorio con la impresora 3D para mantener el antebrazo quieto.

En cuanto al soporte en el movimiento de los dedos, tras hablar con los médicos con los que se colabora para realizar el *serious game* se ha decidido hacer un prototipo que mantenga el antebrazo elevado, estando el codo apoyado en la mesa. Por eso se ha implementado este apoyo provisional hecho con poliuretano y con unas medidas provisionales con las que se pretende conseguir la posición óptima para la rehabilitación del síndrome del túnel carpiano y que pueda ser usado por pacientes de cualquier condición y que ha sido explicado en el apartado de hardware con el mismo nombre.

4.3. PRECISIÓN ÁNGULOS DEDOS

La precisión que se puede obtener con el ángulo de cada una de las falanges proximales y medias de los dedos merece una mención aparte. En este caso se ha realizado con las falanges proximales y con las medias el movimiento de flexión máxima para comprobar el ángulo obtenido.

4.3.1. ÁNGULO MÁXIMO PROXIMALES

En el primer caso se ha hecho el movimiento de doblar las falanges proximales con el fin de ponerlas a más de 90º con la palma obteniendo la siguiente gráfica del valor del ángulo en todo momento.

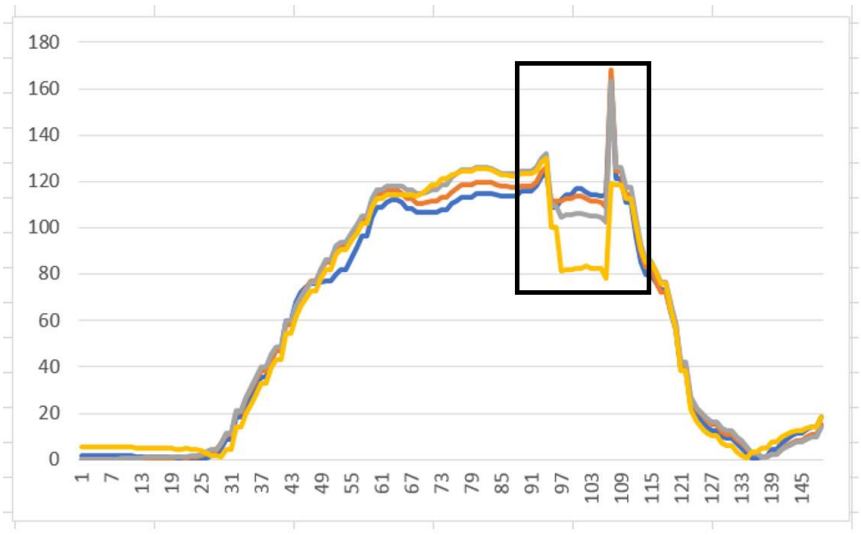


Gráfico 4. 14. Ángulo Máximo Proximales

En esta gráfica están unidos el valor obtenido por cada uno de los dedos y se ha recuadrado el resultado dispar que aparentemente no es correcto. Este resultado es producido por el hecho de que al doblar los dedos llega un punto en el que el *Leap Motion* no detecta bien los dedos por estar algunas falanges superponiéndose a otros huesos.

A la hora de aumentar los grados de giro a más de 90º de las falanges proximales el *Leap Motion* no detecta bien los dedos. Esto hace que en la zona recuadrada en la anterior gráfica el *Leap Motion* interpreta que la posición de la mano sea la siguiente:

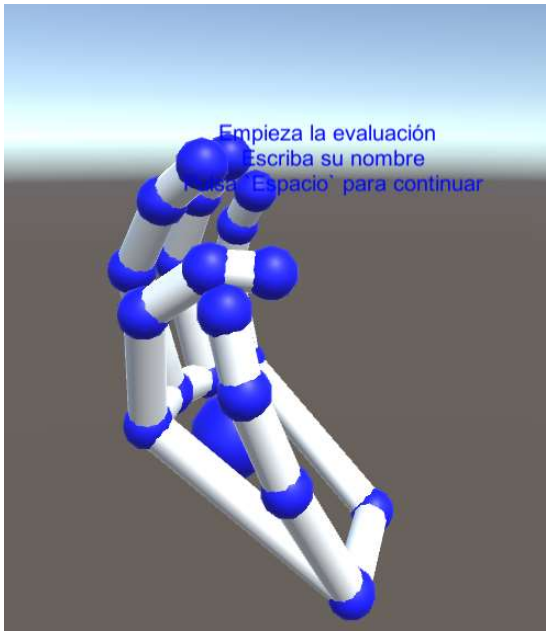


Gráfico 4. 15. Mano a más de 90º

Como se puede intuir esta posición no es la correcta y no es la que el paciente tiene. Por tanto podemos llegar a la conclusión de que este movimiento y los ángulos obtenidos no van a ser tan precisos, al menos al aumentar el ángulo a más de 90º por dar lugar a interpretaciones de la realidad erróneas.

El siguiente caso que analizar ha sido el mismo que el realizado anteriormente con la palma de la mano y los ángulos de extensión máximos obtenidos. Para ello se ha repetido 5 veces el movimiento de flexión de cada uno de los 5 dedos que componen la mano, guardando en cada caso el valor máximo del ángulo de las falanges proximales que se obtiene.

El porcentaje de diferencia que se ha obtenido en cada uno de los dedos es el siguiente:

Dedo pulgar: 63, 51%

Dedo índice: 43, 6%

Dedo corazón: 33, 61%

Dedo anular: 33, 3%

Dedo meñique: 41, 33%

Este porcentaje de diferencia que tiene cada dedo es demasiado grande como para no tenerlo en cuenta. Al haber tanta diferencia los datos obtenidos por las falanges de ángulo de las mismas no pueden ser tomados en cuenta para cuantificar el avance que tiene el paciente. Para poder entender mejor porqué existe tanta disparidad en los resultados en cada repetición hay que analizar la diferencia que tiene la posición que detecta el *Leap Motion* respecto a la real al hacer este movimiento.

4.3.2. POSICIÓN DIGITAL VS POSICIÓN REAL

El alto valor que se ha obtenido en los porcentajes indica que la posición varía en cada caso, esto puede ser posible por tener muy poca precisión en los dedos el *Leap Motion* o porque, aunque siempre que se haga el mismo movimiento dé el mismo resultado, el valor que se obtenga no sea el real, haciendo que una variación de pocos grados en la realidad pueda dar una variación mucho más alta en la posición digital.

Para ver qué es lo que está fallando se ha comparado la posición real con la obtenida de forma digital. Para ello se ha puesto en ángulo de 90º las falanges proximales con la palma como se puede ver en la imagen 4.1.

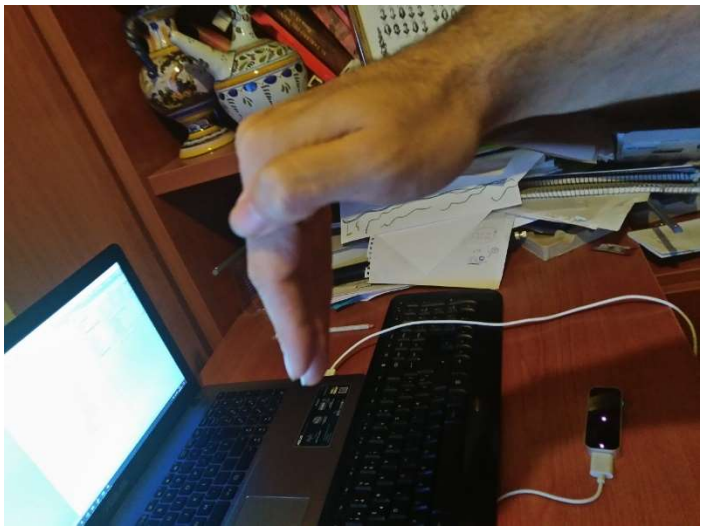


Imagen 4. 1. Posición 90º Mano Real

En este caso el resultado en Unity es el de la figura 4.13.

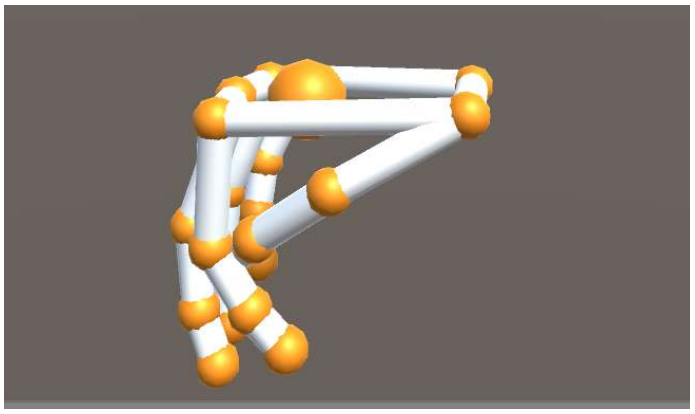


Figura 4. 12. Posición 90º Mano Digital

Pero al variar de forma leve la posición de la mano en la realidad la posición que se obtiene en Unity es la de la figura 4.13.

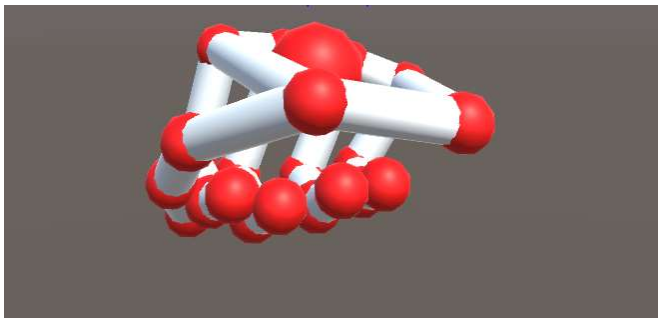


Figura 4. 13. Posición cercana a 90º Mano Digital

Este hecho ocurre en todas las ocasiones dando lugar a la misma posición errónea en la realidad virtual siempre. Este experimento cualitativo nos hace ver que una ligera variación en la realidad puede provocar una gran variación en la virtualidad.

Por último, se ha comprobado si ocurre lo mismo al mover solo una falange media dando en la realidad este resultado:

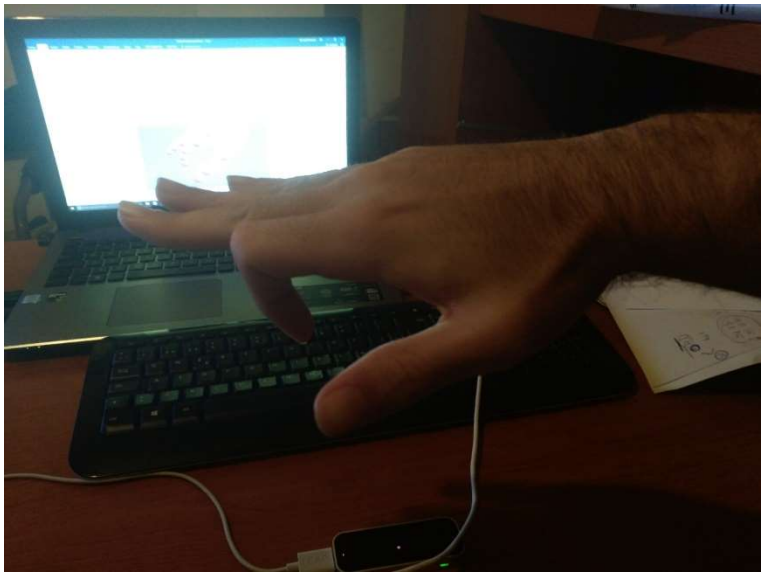


Imagen 4. 2. Posición 90º falange media real

Y en Unity dando el siguiente resultado:

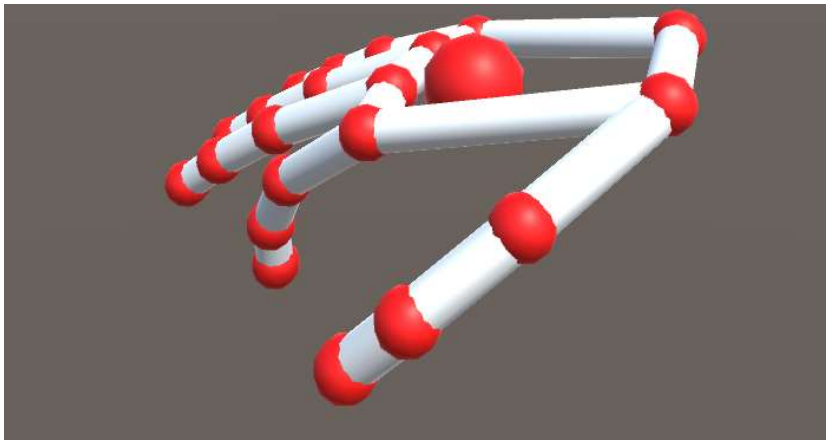


Figura 4. 14. Posición 90º falange media real

Se puede apreciar que la posición que tiene en la realidad no es la misma que tiene en el motor de desarrollo de videojuegos. Considerando esta información se ha llegado a una conclusión sobre cómo actuar en consecuencia a ello explicada en el apartado de "Resultado dedos".

4.3.3. ÁNGULO MÁXIMO MEDIOS

Al hacer el movimiento de "Coger" que se ha explicado anteriormente y capturar el valor en todo momento del ángulo que tiene cada una de las falanges medias de todos los dedos, salvo el pulgar, se consigue la siguiente gráfica.

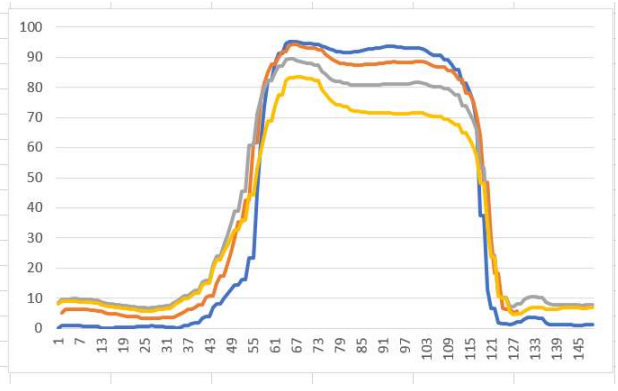


Gráfico 4. 16. Ángulo Máximo Medios

En este caso no se aprecia aparentemente ningún resultado distinto al esperado. Aun así, por todo lo comprobado anteriormente, no puede utilizar estas métricas para cuantificar el avance que tiene un paciente al hacer ese movimiento puesto que los resultados de los ángulos de las falanges no son nada precisos.

4.3.4. RESULTADOS DEDOS

Tras tener en cuenta los resultados explicados anteriormente, se ha decidido modificar la evaluación de los dedos y el juego de los topos, de tal forma que quedan como ha sido explicado a lo largo del proyecto. Para ello en una primera consideración se pensó en validar la posición de coger cuando los dedos de la mano tengan cierto ángulo, mas al comprobar que este ángulo no es preciso se ha decidido que en la evaluación de los dedos se haga el movimiento de coger y el de golpear y capturar para cada paciente que ángulo tiene en cada uno de los movimientos. Con ello se consigue que estas métricas sigan siendo válidas. Si bien el resultado que se obtenga no va a ser cercano a la realidad si se va a conseguir que al menos, siempre que se realice el movimiento de coger, por ejemplo, se tenga el mismo ángulo en la realidad virtual.

Así, en vez de usar el ángulo de las falanges de forma que cuando se consiga cierto ángulo exacto se haga una acción, va a depender de cada paciente. Cuantificando cual es el ángulo obtenido en Unity en cada uno de los dos movimientos de golpear y coger y posteriormente si se llega al mismo ángulo se va a considerar ese movimiento. Es decir, se van a usar los ángulos de las falanges para hacer un booleano específico para cada paciente.

Por todo esto los resultados de los dedos en Unity no se parecen a las reales si bien siguen siendo útiles puesto que, aun así, siempre que la mano en la realidad está en una posición (por ejemplo haciendo un ángulo de 90 º con las proximales) en Unity va a estar en la misma posición (por ejemplo haciendo un ángulo de 68º con las proximales) y gracias a ello se puede usar como un valor lógico de 1 o 0.

4.4. RESULTADOS EVALUACIONES

Los resultados de las evaluaciones son los valores de los ángulos explicados anteriormente. En el caso de la Evaluación Mano los datos obtenidos del ángulo de flexión y extensión y de abducción y aducción pueden servir para saber cuánto mejora el movimiento una persona con el síndrome del túnel carpiano. De esta evaluación solo se saca el ángulo máximo de cada uno de los movimientos, pero al ser guardado en un archivo de Excel cada vez que se realiza se puede llevar un control de cómo ha ido evolucionando cada paciente. Para poder saber hasta que punto los resultados son útiles habría que hacer pruebas con pacientes, que se hará en un futuro próximo para así poder mejorar el proyecto.

En cuanto a la Evaluación Dedos, se saca también el ángulo máximo de ciertos movimientos. En este caso el valor que se obtiene solo sirve para ser utilizado en el juego de los topos puesto que los resultados que se obtienen no son precisos por no ajustarse a la realidad.

4.5. RESULTADOS JUEGOS

En el juego de los topos el resultado obtenido es el tiempo que se tarda den coger y golpear a cada figura. Este resultado sirve para saber en cada sesión si aumenta o disminuye el tiempo que tarda en conseguir la posición que se quiere. Aun así, el tiempo no depende solo de la dolencia del túnel carpiano y por ello hasta que no se pruebe en pacientes no se puede saber la validez de los mismos.

Por otra parte, en el juego de las naves, también se fuerza al paciente a hacer ciertos movimientos que pueden ser beneficiosos para el mismo, aunque en este caso si que los tiempos guardados pueden servir de menos por depender en mayor medida de

otros aspectos como reflejos o habilidad que por la capacidad de mover la mano, que también influye. Por ello también depende de las pruebas que se realicen con los pacientes para dar un juicio de valor del mismo.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado del proyecto se comentan las conclusiones que se sacan tras realizar el proyecto y los posibles trabajos futuros o mejoras que se pueden realizar.

5.1. CONCLUSIONES

Las conclusiones que se pueden sacar de la realización del TFG dependen en gran medida de los objetivos que se pretendían desarrollar. Por ello, como el principal objetivo es crear un *serious game* con el que hacer la rehabilitación del síndrome del túnel carpiano al hacer este trabajo podemos sacar varias conclusiones de ello.

Lo primero que podemos decir es que el sistema de rehabilitación actual no está cuantificado y que depende en gran medida de la opinión de los expertos, que son los que realizan la rehabilitación. Con este tipo de *serious game* y en concreto con la evaluación se puede conseguir apoyar el trabajo del terapeuta ayudándole con información extra del paciente.

Para cumplir el objetivo principal se lleva a cabo la evaluación a principio de cada sesión para que al hacer los juegos sea cada vez más difícil hacer que se mueva la mano o se coja o golpee la manzana y topo. Con esto se consigue un desarrollo progresivo y que el paciente pueda avanzar en cada sesión haciendo que sea esta útil en todo momento.

Por otra parte, aunque el objetivo es ayudar a la rehabilitación del síndrome del túnel carpiano, este tipo de evaluaciones se podría utilizar también para detectar otras

anomalías que tengan que ver con estos movimientos o para saber en qué punto está el túnel carpiano como sistema de detección del síndrome. Para ello habría que ahondar en el trabajo hecho y probar otras finalidades del mismo.

A su vez el añadir tecnología novedosa, hace que los pacientes puedan interesarse más por este tipo de rehabilitación y obtener un *feedback* positivo de ello. También, como es un sistema programado se puede ir modificando en función de los comentarios que se tengan al utilizarse y con ello perfeccionarlo o personalizarlo más.

Al usarse un dispositivo económico y pequeño de fácil uso como es el *Leap Motion* y gracias a haber creado un *serious game* de fácil uso se puede conseguir que el paciente haga los ejercicios no solo en el centro de rehabilitación si no también en casa. Amén de ello puede hacer los ejercicios sin necesidad de estar el terapeuta, lo que realmente implica un abaratamiento del sistema de rehabilitación notable.

En cuanto a las distintas evaluaciones o juegos de los que se compone realmente el *serious game* podemos llegar a varias conclusiones.

Por una parte, respecto a la Evaluación Mano, los valores que obtenemos pueden ser muy útiles y permiten saber en todo momento el ángulo que tiene la palma de la mano, así como la evolución en las sesiones de los mismos. Por ello se puede considerar de gran utilidad dicha evaluación y se puede usar para comprobar otras enfermedades o dolencias que tengan que ver con el movimiento de la muñeca.

La Evaluación Dedos no es precisa y no se pueden sacar conclusiones positivas de la misma, salvo que aun sin ser preciso puede ser usado para forzar al paciente a hacer ciertos movimientos puesto que, aunque no se parezca la posición que se tiene en la realidad con la virtual, sí que tiene cierta conexión dando siempre el mismo valor en el sistema virtual siempre que se tiene una determinada posición en la realidad. Gracias a ello se ha conseguido "capturar" esa posición virtual que se aleja de la realidad y considerarla como si fuera la real y poder usarse en otros juegos como es el de los topos. En definitiva, la métrica obtenida no puede ser usada por los médicos para extraer conclusiones de las mismas, pero sí para usarse en otros juegos.

El juego de las naves obliga al paciente a esforzarse y a llegar a posiciones cercanas a las máximas que puede hacer con la muñeca. Con esto se consigue que se ejercite el túnel carpiano y a su vez que no esté concentrado en ello por tener la

estimulación de estar jugando. Esto puede provocar que el paciente tenga más ganas de hacer la rehabilitación y presente una evolución mayor, puesto que a la hora de tener un estado de salud óptimo no solo es necesario seguir una serie de reglas si no también tener una actitud positiva porque el factor psicológico influye en gran medida en la salud. De ahí la gran importancia de los serious game que pueden conseguir que el paciente esté atento a otros aspectos que no son solo el dolor provocado por forzar los movimientos.

El juego de los topos también tiene la misma finalidad que el juego anterior si bien con otros movimientos que son extraídos de la Evaluación Dedos.

Otra conclusión que se obtiene de hacer el TFG es que la tecnología ha llegado al día a día y está aquí para quedarse. Con proyectos como los que hace *RoboHealth* se puede conseguir mejorar la calidad de vida de las personas menos favorecidas y hay que trabajar en ello para avanzar todo lo posible para usar la tecnología que existe con finalidades humanitarias y para ayudar a las personas que lo necesiten.

También cabe destacar la importancia que tiene la interfaz que se escoja y el aspecto gráfico de la misma. Para conseguir crear un juego serio con el que rehabilitar al paciente no solo hay que fijarse en el dispositivo utilizado y en la utilidad del mismo, sino también en las facilidades que se da al paciente y la sencillez del juego. Un paciente no va a querer jugar a un juego que no tiene un bonito aspecto visual, y que no le motiva. Al tener una dolencia si no le atrae el juego no se va a conseguir el objetivo. Y si no se hace para todas las edades pierde utilidad el mismo.

La siguiente conclusión es que el precio del dispositivo y de la rehabilitación que se obtiene es muy importante y por ello reducir el mismo todo lo posible es una máxima en la que se ha basado este proyecto.

Por último, para poder obtener más conclusiones, tanto positivas como negativas, habría que poner en práctica este *serious game* con pacientes con la dolencia del síndrome del túnel carpiano. Todo esto está pendiente de ser hecho en un futuro cercano.

5.2. FUTUROS TRABAJOS

Este TFG ha dejado multitud de caminos por los que avanzar para conseguir optimizar cómo conseguir cumplir los objetivos del mismo y para conseguir otros objetivos nuevos por estar utilizando una tecnología muy poco utilizada en la que queda todavía mucho por experimentar.

Dentro del *serious game* que se ha hecho se podría aumentar el nivel de especialización para cada paciente regulando la altura a la que aparezcan las manos en la imagen o cambiando los sonidos que tiene y figuras 3D. También se podrían añadir más niveles de dificultad o modificar los mismos.

Todo esto, depende en gran medida del trabajo que se va a llevar a cabo de testearlo con pacientes. Una vez se haga se podrán sacar nuevos trabajos y mejorar sobre las que actuar.

También está pendiente de ser modificado el soporte, puesto que este es solo un prototipo, para que se ajuste de forma correcta a las especificaciones requeridas y pueda ser usado por cualquier paciente.

Por último, hay que hacer mención al hecho de que al usarse un software de código abierto permite ser modificado por cualquier persona con conocimiento de programación y del motor de videojuegos. Una de las modificaciones que se podría hacer es la de utilizar la base del sistema de Evaluación Manos para captar los temblores que tiene una persona con Parkinson u otra enfermedad que provoque temblores en las manos. Con el sistema actualmente utilizado se puede cuantificar el ángulo de la palma de la mano y a partir de este sacar conclusiones del avance de la enfermedad.

6. PRESUPUESTO Y MARCO REGULADOR

6.1. PRESUPUESTO

Dentro de este apartado se incluyen los costes que ha tenido la realización del proyecto. Esto incluye los costes materiales y de personal necesarios para elaborar el todo el trabajo.

6.1.1. COSTE PERSONAL

A continuación, se adjunta una tabla con las horas dedicadas a cada una de las fases del proyecto para la elaboración del presupuesto.

FASES	DÍAS	HORAS/DÍA	HORAS
Estudio	10	6	60
Análisis	6	4	24
Diseño	20	4	80
Desarrollo	40	6	240
Pruebas	10	4	40
Documentación	25	6	150
			TOTAL: 594

Tabla 6. 1. Horas por Fase

Este proyecto ha sido realizado íntegramente por un ingeniero. Al ser un proyecto realizado durante 8 meses sin carácter exclusivo por parte del Ingeniero que lo ha realizado el cálculo del coste de personal se va a hacer en función de las horas, no de los meses trabajados. Para ello se adjunta la siguiente tabla con el coste por hora y carga de trabajo del técnico, así como el coste total sin IVA del personal.

Puesto	Apellidos, Nombre	Tiempo(en horas)	Coste hora	Coste
Ingeniero Electrónico Industrial y Automático	Fernández Barroso, Ángel	594	21, 02 €	12485,88 €

Tabla 6. 2. Coste de personal

El precio por hora del Ingeniero se ha calculado en base a varias ofertas de trabajo consultadas en internet y haciendo un desglose posterior por horas. Como en todo el trabajo la persona encargada de hacerlo ha sido la misma, aun haciendo diferentes tareas se ha considerado el mismo coste por hora. Así también, al comprender en todo momento el trabajo realizado existe un ahorro por parte del cambio de personal que sustituye a la disminución de coste que tendría contratar a personas con distinta cualificación.

6.1.2. COSTE MATERIAL

El coste de material incluye tanto el coste que ha supuesto el software utilizado como el hardware. En el apartado de Software no se ha incluido ninguna tabla por no haber sido considerado ningún coste derivado del mismo. Esto es debido al uso de la versión gratuita de Unity3D y al uso de Visual Studio que es el entorno de programación que viene de serie con el sistema operativo de Windows. Por ello en el coste del Hardware, que incluye el ordenador, viene indirectamente incluido el coste de software de las aplicaciones necesarias para la realización por ser gratuitas o ser incluidas en la adquisición de dicho hardware.

Para calcular el coste de Hardware es necesario tener en cuenta la amortización que estos tienen y para ello hay que seguir la siguiente fórmula con la que se obtiene el coste imputable de cada elemento.

$$\frac{A}{B} \; x \; C \; x \; D$$

Fórmula 1. Coste imputable Hardware

Esta fórmula tiene 4 variables, que son las siguientes:

A: Número de meses de utilización del equipo.

B: Período de depreciación, en este caso se consideran 60 meses.

C: Coste del elemento (sin IVA).

D: Porcentaje de uso del equipo para el proyecto

Con esta fórmula se ha rellenado la siguiente tabla con el coste imputable final del hardware del proyecto.

Descripción	Coste (sin IVA)	Uso dedicado (en %)	Dedicación (meses)	Coste imputable
Ordenador ASUS X550VX	827,54€	100	8	110,34€
Leap Motion	55,34€	100	8	7,38€
				TOTAL: 117,72€

Tabla 6. 3. Coste imputable Hardware

6.1.3. COSTE TOTAL

Por último, se adjunta la tabla con el coste total del proyecto realizado con los costes anteriores.

Descripción	Coste Total
Personal	12485,88€
Hardware	117,72€
Total costes directos	12603,60€
Costes indirectos (20%)	2520,72€
TOTAL	15124,32€

Tabla 6. 4. Coste Proyecto

El coste total del proyecto asciende a **15124,32€**. Dentro de este coste se han incluido los costes indirectos mas no los impuestos indirectos (IVA).

6.2. IMPACTO SOCIO-ECONÓMICO

Este proyecto puede llegar a tener un gran impacto social dentro de la rehabilitación del síndrome del túnel carpiano. Con este *serious game* se pretende mejorar la forma de tratar dicha dolencia así como servir de apoyo a los fisioterapeutas.

El precio que tiene una sesión de fisioterapia aproximado es de 35€. Con este tipo de trabajo se puede conseguir reducir la cantidad de sesiones necesarias por ser estas sustituidas por el trabajo con el *serious game*. También da facilidades al paciente por poder ser usado en cualquier lugar siempre que se tenga un ordenador y el *Leap Motion*. Por tanto, el impacto económico que puede llegar a tener este tipo de trabajos es más que significativo pudiendo reducir en mucho el dinero necesario para rehabilitar. También puede facilitar el trabajo a los fisioterapeutas que pueden cuantificar el avance que tienen los pacientes.

Se podría utilizar a su vez para tratar otras enfermedades en las que también influya el movimiento de la mano si se modifica levemente el código y por tanto aumentar el impacto que produce en la sociedad.

En cuanto al desembolso necesario para poder tener acceso a esta aplicación consiste en sumar el precio del *Leap Motion*, que es de 60 € aproximadamente con el precio que tenga esta aplicación. El dispositivo, al ser necesario su posterior evaluación de los resultados por parte de los médicos, podría ser suministrado durante un tiempo determinado por la clínica y el juego podría ser vendido de muchas formas. También podría ser regalado por cuestiones éticas. Mas si se quiere vender se puede decidir venderlo de forma que la clínica tenga acceso al juego en cualquier ordenador y por tanto poner un precio alto por poder distribuirlo por todos los pacientes de la clínica. Este precio puede ser de 500 € de licencia o poner un precio por paciente u ordenador en el que se use de 2 € para conseguir amortizar el coste de crear dicha aplicación.

A su vez, este proyecto, como entra dentro del proyecto de RoboHealth, tiene como finalidad reducir el cada vez más elevado número de profesionales de los servicios

médicos que actúan en la rehabilitación. Esto es debido al elevado número de personas que precisan de atención médica para rehabilitación médica o cognitiva, que hace que aumente el número de profesionales dedicados a este sector. Por ello, con este trabajo se reduce el impacto económico que representa tener que contar con tal saturación en el sistema médico.

6.3. MARCO REGULADOR

Este proyecto utiliza Unity 3D personal. En este caso la versión no permite su comercialización. Por ello si se quiere acabar realizando una venta de este producto habría que cambiar la versión de Unity. En la versión actual se permite utilizar todos los Assets que se hayan descargado de forma legítima en el juego. Es decir, no hay ningún problema de uso inapropiado del programa. En este caso, como este trabajo es una versión inicial del *serious game* y no ha sido llevado a cabo ningún tipo de venta o comercialización no hay ningún problema.

En el caso de que se quisiera rentar habría que comprar la versión Unity Pro, con la cual si que se puede comercializar los trabajos realizados sin más condiciones que pagar la cuota de usar esta versión. Esto, no llega a ser rentable si el nicho sobre el que se trabaja no es lo suficientemente grande. Por ello existe una segunda alternativa consistente en utilizar Unity ADS. Esta tercera versión del programa permite la libre comercialización de lo que se programe, aunque añade anuncios al juego, que son rentabilizados por Unity [23]. Esa es la "forma de pago" que se tiene por usar este motor de videojuegos de código abierto.

En lo que a la patente del mismo se refiere, se podría comercializar el mismo por parte del alumno porque según la normativa de la UC3M la patente del mismo pertenece a la persona que hace el TFG. Por otra parte, también habría que tener en cuenta que, por norma general, este trabajo se sube a la intranet de la Universidad Carlos III y por tanto habría que desautorizar el ponerlo en el E-Archivo de la Universidad, rellenando lo necesario en el Formulario de Autorización, para evitar así poner el trabajo en abierto.

Este proyecto está pendiente de ser probado con pacientes para comprobar su verdadera utilidad y por ello, como dice la Ley 14/1986, de 25 de abril en el artículo 10 [24], es necesario la previa autorización del paciente y el permiso tanto del médico que esté llevando a cabo su rehabilitación como del centro de salud en el que se encuentre. A su vez, por la Ley General de Sanidad [25], no debe comportar ningún peligro adicional para su salud.

7. BIBLIOGRAFÍA

[1] E.D. Oña, A. Jardon, C. Balaguer. The Automated Box and Blocks Test an Autonomous Assessment Method of Gross Manual Dexterity in Stroke Rehabilitation. 18th Towards Autonomous Robotic Systems (TAROS). Lecture Notes in Artificial Intelligence 10454, ISBN: 978-3-319-64106-5, pages: 101 - 114, Springer. 2017-07-20, Guildford, United Kingdom. 2017.

[2]SINEMBARGO, 27 enero 201). En: SinEmbargo. El Síndrome del Túnel Carpiano afecta al 10 % de la población mundial. http://www.sinembargo.mx/27-01-2015/1229892 [consulta 24 septiembre 2017]

[3]SIGNIFICADOS, s.f. En: Significados. Significado de Hardware. Disponible en: https://www.significados.com/hardware/ [consulta 8 septiembre 2017]

[4]OLEAGA, Jon, 5 febrero 2014. En: ABC. Tecnología. Probamos Leap Motion, el controlador por gestos en 3D para tu ordenador. Disponible en:

http://www.abc.es/tecnologia/informatica/20140204/abci-probamos-leap-motion-201402041411.html [consulta 12 septiembre 2017]

[5] WIKIPEDIA, 5 septiembre 2017. En: Wikipedia. Plug and Play. Disponible en: https://es.wikipedia.org/wiki/Plug_and_play [consulta 12 septiembre 2017]

[6] LEAP MOTION, s.f. En: Leap Motion. Disponible en: https://www.leapmotion.com/ [consulta 24 septiembre 2017]

[7] Wikipedia, 6 abril 2017. En Wikipedia. Era de la información. Disponible en: https://es.wikipedia.org/wiki/Era_de_la_informaci%C3%B3n [consulta 15 mayo 2017]

[8]UNITY, s.f. En: Unity. Requisitos del sistema para la Unity Version. Disponible en: https://unity3d.com/es/unity/system-requirements [consulta 9 septiembre 2017]

[9] UNITY, s.f. En: Unity, Disponible en: https://unity3d.com/es [consulta 20 septiembre 2017]

[10]HORACIO REYES, Alberto, 18 noviembre 2006. En: Efisioterapia. Fisioterapia: pasado, presente y ¿futuro? Disponible en: https://www.efisioterapia.net/articulos/fisioterapia-pasado-presente-y-futuro [consulta 02 septiembre 2017]

[11] BENJAMIN MA, C, 5 septiembre 2015. En: MedlinePlus. Síndrome del túnel carpiano. Disponible en: https://medlineplus.gov/spanish/ency/article/000433.htm [consulta 12 mayo 2017]

[12]WIKIPEDIA, 23 agosto 2017. En: Wikipedia. Juego serio. Disponible en: https://es.wikipedia.org/wiki/Juego_serio [consulta 4 septiembre 2017]

[13]LA VANGUARDIA, 3 julio 2017. En: La Vanguardia. Cultura. Serious Games For Health, los videojuegos diseñados para mejorar la salud. Disponible en: http://www.lavanguardia.com/cultura/20170703/423874892855/serious-games-for-health-videojuegos-salud.html [consulta 5 septiembre 2017]

[14]ORTIZ, Rocío, 3 Julio 2015. En: Unocero. El espectacular simulador de Fórmula 1 de Ferrari. Disponible en: https://www.unocero.com/autos/el-espectacular-simulador-de-formula-1-de-ferrari/ [consulta 20 agosto 2017]

[15]MOTION CAPTURE, 25 febrero 2015. En: Motion Capture. Técnicas de captura. Disponible en: https://coatzayork1.wordpress.com/2015/02/25/tecnicas-de-captura/ [consulta 24 agosto 2017]

[16]INDIEGOGO, s.f. En: Indiegogo. MotionSavvy UNI: 1st sign language to voice system. Disponible en: https://www.indiegogo.com/projects/motionsavvy-uni-1st-sign-language-to-voice-system#/ [consulta 26 agosto 2017]

[17] UNITY, s.f. En: Unity. Tutoriales. Disponible en: https://unity3d.com/es/learn/tutorials [consulta 12 febrero 2017]

[18] LUCICHART, s.f. En: Lucichart. Disponible en: https://www.lucidchart.com/pages/es?utm_source=google&utm_medium=cpc&utm_campaign=lucidchart_spain&gclid=Cj0KCQjwr53OBRCDARIsAL0vKrO_d4Uz2ZckObhUNHzLAZL-NFWR7LypzsBDNsx0nDy-BsN42TX_xcAaAuNMEALw_wcB [consulta 21 septiembre 2017]

[19] ANATOMÍA HUMANA, s.f. En: Anatomía Humana. Los movimientos de la muñeca. Disponible en: http://www.anatomia-humana.com/Movimientos/movimiento-de-la-muneca.html [última consulta 6 septiembre 2017]

[20]GAMELEARN, 6 de marzo de 2017. En: Gamelearn. Todo lo que necesitas saber sobre los serious games y el game-based learning, explicado con ejemplos. Disponible en: https://www.game-learn.com/lo-que-necesitas-saber-serious-games-game-based-learning-ejemplos/ [consulta 4 septiembre 2017]

[21] API LEAP MOTION s.f. En: Leap Motion. Developers Disponible en: https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html [consulta 25 septiembre 2017]

[22] RESPUESTAS.TIPS, 17 mayo 2016. En: Respuestas.Tips. ¿Cómo se llaman los dedos de la mano? Disponible en: http://respuestas.tips/como-se-identifican-a-los-dedos-de-la-mano/ [consulta 9 septiembre 2017]

[23]CVILLEDAMARIN., Junio 2015. En: Taringa. ¿Comercializar juego con unity? Disponible en:

https://www.taringa.net/comunidades/comunidadunity3d/9367320/Comercializar-juego-con-unity.html [consulta 14 septiembre 2017]

[24] BOE, 3 diciembre 2013. En: BOE. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-1986-10499 [consulta 20 septiembre 2017]

[25] OCU, s.f. En: Quién tiene derecho a la sanidad pública. Disponible en: http://www.lapirenaicadigital.es/SITIO/QUIENTIENEDERECHOASANIDADPUBLICA.pdf [consulta 23 septiembre 2017]

AZZAHRAOUI DAOUDI, Omar, 23 febrero 2017. Modificación de estados del teléfono usando la tecnología NFC. Trabajo fin de grado. Leganés: Universidad Carlos III de Madrid [consulta 9 septiembre 2017]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/24280/TFG_Omar_Azzahraoui.pdf?sequence=1&isAllowed=y [consulta 9 septiembre 2017]

MATA SAN JUAN, Henar, 27 febrero 2017. Herramienta de análisis de legibilidad de contenidos educativos. Trabajo fin de grado. Leganés: Universidad Carlos III de Madrid [consulta 9 septiembre 2017]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/24301/TFG_Henar_Mata_SanJuan.pdf?sequence=1&isAllowed=y [consulta 9 septiembre 2017]

CUBEIRO TUIMIL, Cynthia M.ª, octubre 2016. Implementación de aplicaciones "Serious Games" para la evaluación de la destreza manual. Trabajo fin de carrera. Leganés: Universidad Carlos III de Madrid [consulta 9 septiembre 2017].

UC3M, s.f. En: UC3M. Biblioteca. Disponible en: .https://www.uc3m.es/ss/Satellite/Biblioteca/es/TextoMixta/1371214019341/Derechos_de_explotacion [consulta 16 septiembre 2017]

JUÁREZ JIMÉNEZ, Fátima I., s.f. En: El resumen, el abstract y la síntesis. Disponible en: https://es.slideshare.net/FatyJuarezJ/el-resumen-el-abstract-y-la-sntesis [consulta 20 agosto 2017]

FERRER, Jesús, 2010. En: metodología02. Tipos de investigación y diseño de investigación. Disponible en:

http://metodologia02.blogspot.com.es/p/operacionalizacion-de-variables.html [consulta 13 septiembre 2017]

API UNITY, s.f. En: Unity. Documentation. Disponible en: https://docs.unity3d.com/ScriptReference/ [consulta 25 septiembre 2017]

WIKIPEDIA, 7 mayo 2017. En: Wikipedia. Software. Disponible en: https://es.wikipedia.org/wiki/Software [consulta 13 septiembre 2017]

8. ANEXO

El compendio de anexos que incluye este Trabajo Fin de Grado está relacionado con el código empleado para la elaboración de la aplicación, A continuación se detalla el mismo.

8.1. MENU_INICIAL

8.1.1. INICIO SCRIPT

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.IO;

public class InicioScript : MonoBehaviour {


    public Canvas menuInicialCanvas;
    public Canvas jugadorCanvas;
    public Canvas crearJugadorCanvas;
    public Canvas juegosCanvas;
    public Canvas cargarJugadorCanvas;
    public Text manoText;
    public Text manoLesionadaText;
    public CargarJugadorScript cargarJugadorScript;

    // Use this for initialization
    void Start ()
    {
        jugadorCanvas.enabled = false;
        crearJugadorCanvas.enabled = false;
        juegosCanvas.enabled = false;
        cargarJugadorCanvas.enabled = false;
        manoText.text = "";
        string volver;
```

```
        volver =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Volver/Volver.doc";
        if (File.Exists(volver))
        {
            StreamReader read = File.OpenText(volver);
            string leer;
            leer = read.ReadLine().ToString();

            if (leer != "a")
            {
                menuInicialCanvas.enabled = false;
                juegosCanvas.enabled = true;
            }
            read.Close();

        }

    }

    public void InicioButton()
    {
        jugadorCanvas.enabled = true;
        menuInicialCanvas.enabled = false;
        string guardadoNombre;
        guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
        StreamWriter write2 = File.CreateText(guardadoNombre);
        write2.Write("b");
        write2.Flush();
        write2.Close();
        string guardadoVolver;
        guardadoVolver =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Volver/Volver.doc";
        StreamWriter write = File.CreateText(guardadoVolver);
        write.Write("b");
        write.Flush();
        write.Close();

    }

    public void InicioJuntoButton()
    {
        jugadorCanvas.enabled = true;
        menuInicialCanvas.enabled = false;

        string guardadoNombre;
        guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
        StreamWriter write2 = File.CreateText(guardadoNombre);
        write2.Write("a");
        write2.Flush();
        write2.Close();

        string guardadoVolver;
        guardadoVolver =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Volver/Volver.doc";
        StreamWriter write = File.CreateText(guardadoVolver);
        write.Write("b");
        write.Flush();
        write.Close();


    }

    public void ExitButton()
    {

        string guardadoVolver;
        guardadoVolver =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Volver/Volver.doc";

        StreamWriter write2 = File.CreateText(guardadoVolver);
        write2.Write("a");
        write2.Flush();
```

```
        write2.Close();
        Application.Quit();
    }

    public void CrearJugadorButton()
    {
        jugadorCanvas.enabled = false;
        crearJugadorCanvas.enabled = true;
    }

    public void CargarJugadorButton()
    {
        jugadorCanvas.enabled = false;
        cargarJugadorCanvas.enabled = true;
    }

    public void GuardarJugadorButton()
    {
        string seguido;
        seguido =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
        if (File.Exists(seguido))
        {
            StreamReader read = File.OpenText(seguido);
            string leer;
            leer = read.ReadLine().ToString();

            if (leer == "a")
            {
                SceneManager.LoadScene("Prueba1");

            }
            else
            {

                crearJugadorCanvas.enabled = false;
                juegosCanvas.enabled = true;
                manoText.text = "MANO: " + manoLesionadaText.text;
            }
            read.Close();
        }

        else
        {
            crearJugadorCanvas.enabled = false;
            juegosCanvas.enabled = true;
            manoText.text = "MANO: " + manoLesionadaText.text;
        }

    }

    public void CancelarJugadorButton()
    {
        crearJugadorCanvas.enabled = false;
        menuInicialCanvas.enabled = true;
    }

    public void ComprobarJugadorButton()
    {
        string seguido;
        seguido =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
        if (File.Exists(seguido))
        {
            StreamReader read = File.OpenText(seguido);
            string leer;
            leer = read.ReadLine().ToString();

            if (leer == "a")
            {
                SceneManager.LoadScene("Prueba1");

            }
            else
```

```
            {

                cargarJugadorCanvas.enabled = false;
                juegosCanvas.enabled = true;
                manoText.text = "";
            }
        }
        else
        {
            cargarJugadorCanvas.enabled = false;
            juegosCanvas.enabled = true;
            manoText.text = "";
        }
    }

    public void CancelarCargarJugadorButton()
    {
        cargarJugadorCanvas.enabled = false;
        menuInicialCanvas.enabled = true;
    }

    public void EvaluacionMano()
    {
        SceneManager.LoadScene("Prueba1");
    }

    public void EvaluacionDedo()
    {
        SceneManager.LoadScene("Prueba2");
    }

    public void JugarNaves()
    {
        SceneManager.LoadScene("Main");
    }
    public void JugarTopos()
    {
        SceneManager.LoadScene("Topos");
    }

    public void AtrasButton()
    {
        juegosCanvas.enabled = false;
        menuInicialCanvas.enabled = true;
    }
}
```

8.1.2. CREAR JUGADOR SCRIPT

```
using UnityEngine;
using System.IO;
using UnityEngine.UI;

public class CrearJugadorScript : MonoBehaviour {


    public Text nombreText;
    public Text motivoText;
    public Text manoLesionadaText;
    public InicioScript inicioScript;
    string manoLesionada;
    public void GuardarButton()
    {
        {
            if ( nombreText.text != "" && motivoText.text != "" && manoLesionadaText.text !=
"" && (manoLesionadaText.text =="derecha" || manoLesionadaText.text =="izquierda"))
            {
                string guardado;
                guardado =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado/"  + nombreText.text +
".csv";
                if (manoLesionadaText.text == "izquierda")
```

```
                {
                    manoLesionada = "Izquierda";
                }
                if (manoLesionadaText.text == "derecha")
                {
                    manoLesionada = "Derecha";
                }


                if (!File.Exists(guardado))
                {
                    //Esto es si queremos crear un archivo nuevo
                    StreamWriter write = File.CreateText(guardado);
                    write.Write("\n , DATOS PACIENTE" +
                        "\n , Nombre y Apellidos: ," + nombreText.text
                       + "\n , Motivo: , " + motivoText.text
                       + "\n , Mano Lesionada: , " + manoLesionada
                       + "\n , , , , , , ANGULOS MANO , , , , , , ANGULOS PROXIMALES DEDOS ,
, , , , , ANGULOS MEDIOS DEDOS\n "
                       + ", , , , Fecha , Maximo angulo vertical , Minimo angulo vertical ,
Maximo angulo horizontal , Minimo angulo vertical , , Fecha, Maximo angulo proximal indice,
Maximo angulo proximal corazon, Maximo angulo proximal anular, Maximo angulo proximal
meñique"
                       + ", ,  Maximo angulo medio indice, Maximo angulo medio corazon,
Maximo angulo medio anular, Maximo angulo medio menique , , Tiempo Manzana , , Tiempo Topo ,
, Fallos \n");
                    write.Flush();
                    write.Close();
                }

                else
                {
                    StreamWriter write = File.AppendText(guardado);
                    write.Write(
                         "\n \n \n , DATOS PACIENTE Actualizados"
                       + "\n , Nombre y Apellidos: ," + nombreText.text
                       + "\n , Motivo: , " + motivoText.text
                       + "\n , Mano Lesionada: , " + manoLesionadaText.text
                       + "\n , , , , , , ANGULOS MANO , , , , , , ANGULOS PROXIMALES DEDOS ,
, , , , , ANGULOS MEDIOS DEDOS\n "
                       + ", , , , Fecha , Maximo angulo vertical , Minimo angulo vertical ,
Maximo angulo horizontal , Minimo angulo vertical , , Fecha, Maximo angulo proximal indice,
Maximo angulo proximal corazon, Maximo angulo proximal anular, Maximo angulo proximal
meñique"
                       + ", ,  Maximo angulo medio indice, Maximo angulo medio corazon,
Maximo angulo medio anular, Maximo angulo medio menique , , Tiempo Manzana , , Tiempo Topo
\n");
                    write.Flush();
                    write.Close();
                }

                string guardadoNombre;

                guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Nombre/Nombre.doc";
                StreamWriter write2 = File.CreateText(guardadoNombre);
                write2.Write(nombreText.text + "\n" + manoLesionada);

                write2.Flush();
                write2.Close();
                inicioScript.GuardarJugadorButton();
            }
        }
    }
}
```

8.1.3. CARGAR JUGADOR SCRIPT

```
using UnityEngine;
using System.IO;
using UnityEngine.UI;
```

```
public class CargarJugadorScript : MonoBehaviour {


    public Text nombreCargarText;
    public InicioScript inicioScript;
    string manoMala;
    public Text pruebaText;

    public void ComprobarButton()
    {
        {
            if (nombreCargarText.text != "" )
            {
                string guardado;
                guardado =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado/" + nombreCargarText.text
+ ".csv";

                if (File.Exists(guardado))
                {
                    inicioScript.ComprobarJugadorButton();
                    char[] leer = new char[50];

                    StreamReader read = File.OpenText(guardado);
                    read.ReadLine().ToCharArray();
                    read.ReadLine().ToCharArray();
                    read.ReadLine().ToCharArray();
                    leer = read.ReadToEnd().ToCharArray();

                    //The char 38 is where is the D or the I of Derecha and Izquierda
                    if (leer[37] == 'D' || leer[38] == 'D' || leer[39] =='D' || leer[40]
=='D' || leer[41] == 'D' || leer[42] == 'D')
                    {
                        manoMala = "Derecha";
                    }

                    else if (leer[37] == 'I' || leer[38] == 'I' || leer[39] == 'I' ||
leer[40] == 'I' || leer[41] == 'I' || leer[42] == 'I')
                    {
                        manoMala = "Izquierda";
                    }

                    else
                    {
                        manoMala = "Ambas";
                    }

                    read.Close();
                    string guardadoNombre;

                    guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Nombre/Nombre.doc";
                    StreamWriter write2 = File.CreateText(guardadoNombre);
                    write2.Write(nombreCargarText.text + "\n" + manoMala );
                    write2.Flush();
                    write2.Close();
                    pruebaText.text = "MANO: " + manoMala;
                }
            }

        }
    }

    public string ManoLesionada()
    {
        return manoMala;
    }
}
```

8.2. PRUEBA 1

8.2.1. BOTONES

```
using UnityEngine;
namespace Leap.Unity
{
    public class Botones : MonoBehaviour
    {

        public Evaluation evaluation;

        public void EmpezarButton()
        {
            evaluation.EmpezarButton();
        }

        public void ContinuarButton()
        {
            evaluation.ContinuarButton();
        }

        public void FinalizarButton()
        {
            evaluation.FinalizarButton();
        }

        public void GuardarButton()
        {
            evaluation.GuardarButton();
        }

        public void ReiniciarButton()
        {
            evaluation.ReiniciarButton();
        }
        public void ContinuarExplicacionButton()
        {
            evaluation.ContinuarExplicacionButton();
        }

        public void FinalizarExplicacionButton()
        {
            evaluation.FinalizarExplicacionButton();
        }
    }
}
```

8.2.2. EVALUACIÓN

```
using System;
using UnityEngine;
using UnityEngine.UI;
using Leap.Unity.Attributes;
using System.IO;
using UnityEngine.SceneManagement;

namespace Leap.Unity
{
    public class Evaluation : MonoBehaviour
    {

        [Units("seconds")]
        [Tooltip("The interval in seconds at which to check this detector's conditions.")]
        [MinValue(0)]
        public float Period = .1f; //seconds

        [Tooltip("The hand model to watch. Set automatically if detector is on a hand.")]
        public IHandModel HandModelRight = null;
```

```
public IHandModel HandModelLeft = null;
IHandModel HandModel = null;

Bone probando;
Vector3 probandoVector;
float probandoAngle;

Hand hand;

Vector3 palmVector;
Vector3 palmVector2;
Vector2 palma;
Vector2 palma2;
Vector2 horizonte;
Vector2 horizonte2;
float angle;
float angle2;



public Text finalText;
public Text nombreText;
public GameObject horizontalLine;
public GameObject verticalLine;

public Text marcasText;

private bool inicio;
private bool horizontal;
private bool vertical;
private bool final;

private int flagUp;
private int flagDown;

float horizontalMax, verticalMax, horizontalMin, verticalMin;

DateTime fecha;

public Canvas empezarCanvas;
bool empezarBool;

public Canvas continuarCanvas;
bool continuarBool;

public Canvas finalizarCanvas;
bool finalizarBool;

public Canvas guardarCanvas;
bool guardarBool;
bool reiniciarBool;

public Canvas continuarExplicacionCanvas;

public Canvas finalizarExplicacionCanvas;
string Nombre;
string NombreMano;


void Start()
{

    //We said that we are not in any state
    inicio = true;
    horizontal = false;
    vertical = false;
    final = false;



    finalText.text = "";
    marcasText.text = "";

    //the values are 0 when we start
```

```
    horizontalMax = 0;
    horizontalMin = 0;
    verticalMax = 0;
    verticalMin = 0;
    angle = 0;
    angle2 = 0;
    flagUp = 0;
    flagDown = 0;
    horizontalLine.SetActive(false);
    verticalLine.SetActive(false);

    empezarCanvas.enabled = true;
    empezarBool = false;
    continuarCanvas.enabled = false;
    continuarBool = false;
    finalizarCanvas.enabled = false;
    finalizarBool = false;
    guardarCanvas.enabled = false;
    guardarBool = false;
    reiniciarBool = false;
    continuarExplicacionCanvas.enabled = false;
    finalizarExplicacionCanvas.enabled = false;



    Cargar();

    if (NombreMano == "Izquierda" )
    {
        HandModel = HandModelLeft;
    }

    else if (NombreMano == "Derecha")
    {
        HandModel = HandModelRight;
    }

    marcasText.text = NombreMano;
}

void Update()
{

    if (HandModel != null)
    {
        hand = HandModel.GetLeapHand();

        if (hand != null)
        {
            palmVector = hand.PalmNormal.ToVector3();
            palmVector2 = hand.Direction.ToVector3();
            horizonte.Set(1, 0);
            palma.Set(palmVector.z, palmVector.y);
            angle = (Vector2.Angle(horizonte, palma) - 90) * -1;
            horizonte2.Set(1, 0);
            palma2.Set(palmVector2.x, palmVector2.z);
            angle2= (Vector2.Angle(palma2, horizonte2)-90);
        }
    }

    if (inicio)
    {


        if (empezarBool)
        {
            inicio = false;
            horizontal = true;

            horizontalLine.SetActive(true);
            empezarBool = false;
        }
    }
```

```
            if (horizontal)
            {


                //We grab the horizontalMax if it is bigger that the last horizontalMax
                if (angle >= 0 && angle > horizontalMax)
                {
                    horizontalMax = angle;
                    flagUp = 1;
                }

                //We grab the horizontalMin if it is smaller that the last horizontalMin
                if (angle < 0 && angle < horizontalMin)
                {
                    horizontalMin = angle;
                    flagDown = 1;
                }

                //If we pulse H we go to vertical
                if (continuarBool && flagUp == 1 && flagDown == 1)

                {
                    horizontal = false;
                    vertical = true;

                    verticalLine.SetActive(true);

                    continuarBool = false;
                }

                float movimientoAngle;
                movimientoAngle = (angle * 0.2f / 90);

                if (angle > 0)
                {
                    horizontalLine.GetComponent<Transform>().position = new Vector3(0,
movimientoAngle, 0.4821f);
                }

                else
                {
                    horizontalLine.GetComponent<Transform>().position = new Vector3(0,
movimientoAngle, 0.4821f);
                }
            }

            //the state vertical
            if (vertical)
            {


                //We grab the horizontalMax if it is bigger that the last horizontalMax
                if (angle2 >= 0 && angle2 > verticalMax)
                {
                    verticalMax = angle2;
                    flagUp = 2;
                }

                //We grab the horizontalMin if it is smaller that the last horizontalMin
                if (angle2 < 0 && angle2 < verticalMin)
                {
                    verticalMin = angle2;
                    flagDown = 2;
                }

                //If we pulse V we go to final
                if (finalizarBool && flagUp == 2 && flagDown == 2)
                {
                    vertical = false;
                    final = true;

                    Destroy(verticalLine);
                    finalizarBool = false;
                }
```

```
                float movimientoAngle2;
                movimientoAngle2 = (angle2 * 0.2f / 90);

                if (angle2 > 0)
                {
                    verticalLine.GetComponent<Transform>().position = new Vector3(-
movimientoAngle2, 0, 0.4821f);
                }
                else
                {
                    verticalLine.GetComponent<Transform>().position = new Vector3(-
movimientoAngle2, 0, 0.4821f);
                }
            }

            //the sate final
            if (final)
            {
                finalText.text = "¡ENHORABUENA! \n Ya has acabado la evaluación";
                marcasText.text = "TUS RESULTADOS \n " +
                    "Subida: " + horizontalMax
                    + "\n Bajada: " + horizontalMin
                    + "\n Derecha: " + verticalMin
                    + "\n Izquierda: " + verticalMax;

                Destroy(verticalLine);

                if (reiniciarBool)
                {
                    final = false;
                    reiniciarBool = false;

                    finalText.text = "";
                    SceneManager.LoadScene("Prueba1");
                }

                //If we pulse S we save the scene
                if (guardarBool)
                {
                    final = false;
                    guardarBool = false;
                    finalText.text = "";

                    Guardar();
                    string seguido;
                    seguido =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
                    StreamReader read = File.OpenText(seguido);
                    string leer;
                    leer = read.ReadLine().ToString();

                    if (leer == "a")
                    {
                        SceneManager.LoadScene("Prueba2");

                    }
                    else
                    {
                        SceneManager.LoadScene("Menu_Inicial");
                    }

                }
            }
        }

        void Cargar()
        {
            string guardadoNombre;
            guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Nombre/Nombre.doc";
            StreamReader read = File.OpenText(guardadoNombre);
            Nombre = read.ReadLine();
            NombreMano = read.ReadLine();
            read.Close();
```

```
        }


        void Guardar()
        {
            fecha = DateTime.Now;

            string guardado;

            guardado = "C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado/"
+ Nombre + ".csv";

                StreamWriter write = File.AppendText(guardado);
                write.Write( ", , , ," +  fecha.ToString() + "," + horizontalMax + "," +
horizontalMin + "," + verticalMax + "," + verticalMin + "\n");

                write.Flush();
                write.Close();

            string guardado2;
            guardado2 =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado_angulos_muneca\\"
+ Nombre + ".doc";

            StreamWriter write2 = File.CreateText(guardado2);
            write2.Write(horizontalMax + "\n" + horizontalMin + "\n" + verticalMax + "\n" +
verticalMin);
            write2.Flush();
            write2.Close();
        }

        public void EmpezarButton()
        {

            empezarCanvas.enabled = false;
            continuarExplicacionCanvas.enabled = true;
        }

        public void ContinuarButton()
        {
            if (flagUp == 1 && flagDown == 1)
            {
                continuarCanvas.enabled = false;
            finalizarExplicacionCanvas.enabled = true;
            Destroy(horizontalLine);
            }

        }

        public void FinalizarButton()
        {
            if (flagUp == 2 && flagDown == 2)
            {
                finalizarBool = true;
                finalizarCanvas.enabled = false;
                guardarCanvas.enabled = true;
            }
        }

        public void GuardarButton()
        {
            guardarBool = true;
            guardarCanvas.enabled = false;
        }

        public void ReiniciarButton()
        {
            reiniciarBool = true;
            guardarCanvas.enabled = false;
        }
        public void ContinuarExplicacionButton()
        {
            continuarExplicacionCanvas.enabled = false;
            continuarCanvas.enabled = true;
```

```
            empezarBool = true;

        }

        public void FinalizarExplicacionButton()
        {
            finalizarExplicacionCanvas.enabled = false;
            finalizarCanvas.enabled = true;
            continuarBool = true;

        }
    }
}
```

8.3. PRUEBA 2

8.3.1. BOTONES 2

```
using UnityEngine;
namespace Leap.Unity
{
    public class Botones2 : MonoBehaviour
    {

        public Evaluation2 evaluation;

        // Use this for initialization
        void Start()
        {

        }

        // Update is called once per frame
        void Update()
        {

        }

        public void EmpezarButton()
        {
            evaluation.EmpezarButton();
        }

        public void ContinuarButton()
        {
            evaluation.ContinuarButton();
        }

        public void FinalizarButton()
        {
            evaluation.FinalizarButton();
        }

        public void GuardarButton()
        {
            evaluation.GuardarButton();
        }

        public void ReiniciarButton()
        {
            evaluation.ReiniciarButton();
        }

        public void ContinuarExplicacionButton()
        {
            evaluation.ContinuarExplicacionButton();
        }

        public void FinalizarExplicacionButton()
```

```
        {
            evaluation.FinalizarExplicacionButton();
        }

    }
}
```

8.3.2. EVALUACION 2

```
using System;
using UnityEngine;
using UnityEngine.UI;
using Leap.Unity.Attributes;
using System.IO;
using UnityEngine.SceneManagement;

namespace Leap.Unity
{
    public class Evaluation2 : MonoBehaviour
    {

        [Units("seconds")]
        [Tooltip("The interval in seconds at which to check this detector's conditions.")]
        [MinValue(0)]
        public float Period = .1f; //seconds

        [Tooltip("The hand model to watch. Set automatically if detector is on a hand.")]
        public IHandModel HandModelRight = null;
        public IHandModel HandModelLeft = null;
        IHandModel HandModel = null;

        Hand hand;



        Vector2 bone;
        Vector2 horizonte;




        public Text finalText;
        public Text marcasText;
        public Text resultadosText;

        // the bools are as states
        private bool inicio;
        private bool proximalBool;
        private bool medioBool;
        private bool saveProximalBool;
        private bool saveMedioBool;
        private bool final;

        Finger[] prueba = new Finger[5];
        Vector3[] a = new Vector3[5];
        Bone[] proximal = new Bone[5];
        Bone[] medio = new Bone[5];
        Vector3[] proximalVector = new Vector3[5];
        Vector3[] medioVector = new Vector3[5];
        float[] proximalAngle = new float[5];
        float[] medioAngle = new float[5];

        float[] maxProximalAngle = new float[5];

        float[] maxMedioAngle = new float[5];
```

```
DateTime fecha;

public Canvas empezarCanvas;
bool empezarBool;

public Canvas continuarCanvas;
bool continuarBool;

public Canvas finalizarCanvas;
bool finalizarBool;

public Canvas continuarExplicacionCanvas;

public Canvas finalizarExplicacionCanvas;

public Canvas guardarCanvas;
bool guardarBool;
bool reiniciarBool;



//he añadido new a la variable name

bool loadBool;


string Nombre;
string NombreMano;


void Start()
{

    //We said that we are not in any state
    inicio = true;
    proximalBool = false;
    medioBool = false;
    final = false;

    //We said that we are not in any state

    finalText.text = "";
    marcasText.text = "";
    resultadosText.text = "";

    for(int i =1; i<5; i++)
    {
        maxMedioAngle[i] = 0;
        maxProximalAngle[i] = 0;
    }

    empezarCanvas.enabled = true;
    empezarBool = false;
    continuarCanvas.enabled = false;
    continuarBool = false;
    finalizarCanvas.enabled = false;
    finalizarBool = false;
    guardarCanvas.enabled = false;
    guardarBool = false;
    reiniciarBool = false;
    continuarExplicacionCanvas.enabled = false;
    finalizarExplicacionCanvas.enabled = false;

    Cargar();

    if (NombreMano == "Izquierda")
    {
        HandModel = HandModelLeft;

    }
    else if (NombreMano == "Derecha")
    {
```

```
        HandModel = HandModelRight;

    }
    // StartCoroutine(Evaluate());
}


void Update()
{


    if (HandModel != null)

    {
        hand = HandModel.GetLeapHand();
        if (hand != null)
        {



            for (int i = 0; i < 5; i++)
            {
                prueba[i] = hand.Fingers[i];
                horizonte.Set(1, 0);

                if (i > 0)
                {
                    proximal[i] = prueba[i].bones[1];
                    proximalVector[i] = proximal[i].Direction.ToVector3();
                    bone.Set(proximalVector[i].z, proximalVector[i].y);
                    proximalAngle[i] = (Vector2.Angle(horizonte, bone) );
                    medio[i] = prueba[i].bones[2];
                    medioVector[i] = medio[i].Direction.ToVector3();
                    bone.Set(medioVector[i].z, medioVector[i].y);
                    medioAngle[i] = (Vector2.Angle(horizonte, bone) ) ;
                }
            }
        }
    }

    if (inicio)
    {

        if (empezarBool)
        {
            inicio = false;
            proximalBool = true;
            empezarBool = false;
        }
    }

    if (proximalBool)
    {


        // Thread.Sleep(1000); Esto era de prueba para saber como esperar n segundos
        for(int i=1; i<5; i++)
        {
            if (proximalAngle[i] >= 0 && proximalAngle[i] > maxProximalAngle[i])
            {
                maxProximalAngle[i] = proximalAngle[i];
            }
        }

        if (continuarBool )
        {
            proximalBool = false;
            medioBool = true;
            continuarBool = false;
        }
    }

    if (medioBool)
```

```
            {
                for (int i = 1; i < 5; i++)
                {
                    if (medioAngle[i] >= 0 && medioAngle[i] > maxMedioAngle[i])
                    {
                        maxMedioAngle[i] = medioAngle[i];
                    }
                }

                if (finalizarBool)
                {
                    medioBool = false;
                    final = true;
                    finalizarBool = false;
                }
            }

            //the sate final
            if (final)
            {
                finalText.text = "¡ENHORABUENA! \n Ya has acabado la evaluación";
                resultadosText.text = "TUS RESULTADOS";
                marcasText.text = "FALANGES PROXIMALES: \n " +
                    "Índice Proximal: " + maxProximalAngle[1]
                    + "\n Corazón Proximal: " + maxProximalAngle[2]
                    + "\n Anular Proximal: " + maxProximalAngle[3]
                    + "\n Meñique Proximal: " + maxProximalAngle[4]
                    + "\n  FALANGES MEDIAS: \n " +
                    "Índice Media: " + maxMedioAngle[1]
                    + "\n Corazón Media: " + maxMedioAngle[2]
                    + "\n Anular Media: " + maxMedioAngle[3]
                    + "\n Meñique Media: " + maxMedioAngle[4];

                if (reiniciarBool)
                {
                    final = false;
                    finalText.text = "";
                    reiniciarBool = false;
                    SceneManager.LoadScene("Prueba2");
                }

                if (guardarBool)
                {
                    final = false;
                    finalText.text = "";
                    guardarBool = false;

                    Guardar();
                    GuardarSave();
                    string seguido;
                    seguido =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
                    StreamReader read = File.OpenText(seguido);
                    string leer;
                    leer = read.ReadLine().ToString();

                    if (leer == "a")
                    {
                        SceneManager.LoadScene("Main");

                    }
                    else
                    {
                        SceneManager.LoadScene("Menu_Inicial");
                    }

                }
            }
        }

        void Cargar()
        {
            string guardadoNombre;
```

```
            guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Nombre/Nombre.doc";
            StreamReader read = File.OpenText(guardadoNombre);
            Nombre = read.ReadLine();
            NombreMano = read.ReadLine();
            read.Close();
        }


        void Guardar()
        {
            fecha = DateTime.Now;
            string guardado;
            guardado = "C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado/"
+ Nombre + ".csv";

            StreamWriter write = File.AppendText(guardado);
            write.Write(", , , , , , , , , ," + fecha.ToString() + "," + maxProximalAngle[1]
+ "," + maxProximalAngle[2] + "," + maxProximalAngle[3] + "," + maxProximalAngle[4] +
                ", ," + maxMedioAngle[1] + "," + +maxMedioAngle[2] + "," + maxMedioAngle[3]
+ "," + maxMedioAngle[4] + "\n");
            write.Flush();
            write.Close();
        }

        void GuardarSave()
        {
            string guardadoNombre;
            guardadoNombre =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_angulos_dedos/" + Nombre
+ ".doc";

            StreamWriter write2 = File.CreateText(guardadoNombre);
            write2.Write(maxProximalAngle[1] + "\n" + maxProximalAngle[2] + "\n" +
maxProximalAngle[3] + "\n" + maxProximalAngle[4] + "\n" + maxMedioAngle[1] + "\n" +
maxMedioAngle[2] + "\n" + maxMedioAngle[3] + "\n" + maxMedioAngle[4] );
            write2.Flush();
            write2.Close();
        }

        public void EmpezarButton()
        {

            empezarCanvas.enabled = false;
            continuarExplicacionCanvas.enabled = true;
        }

        public void ContinuarButton()
        {

            continuarCanvas.enabled = false;
            finalizarExplicacionCanvas.enabled = true;
        }

        public void FinalizarButton()
        {
            finalizarBool = true;
            finalizarCanvas.enabled = false;
            guardarCanvas.enabled = true;
        }

        public void GuardarButton()
        {
            guardarBool = true;
            guardarCanvas.enabled = false;
        }

        public void ReiniciarButton()
        {
            reiniciarBool = true;
            guardarCanvas.enabled = false;
        }

        public void ContinuarExplicacionButton()
```

```
        {
            continuarExplicacionCanvas.enabled = false;
            continuarCanvas.enabled = true;
            empezarBool = true;

        }

        public void FinalizarExplicacionButton()
        {
            finalizarExplicacionCanvas.enabled = false;
            finalizarCanvas.enabled = true;
            continuarBool = true;

        }
    }
}
```

8.4. MAIN

8.4.1. BOTONES NAVES

```
using UnityEngine;


namespace Leap.Unity
{
    public class BotonesNaves : MonoBehaviour
    {

        public GameController gameController;


        public void EmpezarButton()
        {
            gameController.EmpezarButton();
        }

        public void ReiniciarButton()
        {
            gameController.ReiniciarButton();
        }

        public void SalirButton()
        {
            gameController.SalirButton();
        }

    }
}
```

8.4.2. DESTROY BY BOUNDARY

```
using UnityEngine;

public class Destroy_by_Boundary : MonoBehaviour {

    void OnTriggerExit(Collider other)
    {
        // Destroy everything that leaves the trigger
        Destroy(other.gameObject);
    }
}
```

8.4.3. DESTROY BY CONTACT

```
using UnityEngine;
```

```
public class Destroy_By_Contact : MonoBehaviour {

    public GameObject explosion;
    public GameObject playerexplosion;
    public int scoreValue;

    private Leap.Unity.GameController gameController;


    void Start()
    {
        GameObject gameControllerObject = GameObject.FindWithTag ("GameController");
        if (gameControllerObject != null)
        {
            gameController = gameControllerObject.GetComponent<Leap.Unity.GameController>();
        }
        if (gameController == null)
        {
            Debug.Log("Cannot find `GameControler`script");
        }
    }

    void OnTriggerEnter(Collider other)
    {
        //Debug.Log(other.name);

        if (other.tag == "Boundary")
        {

            return;
        }

        if (other.tag == "Player")
        {
            gameController.Asteroide_Fallar();
          //  Instantiate(playerexplosion, transform.position, transform.rotation);
         //   gameController.GameOver();

        }

        if (other.tag == "Bolt")
        {
            gameController.Asteroide_Destruir();
            Destroy(other.gameObject);
        }


        Instantiate(explosion, other.transform.position, other.transform.rotation);

        gameController.AddScore(scoreValue);

        Destroy(gameObject);

    }
}
```

8.4.4. GAME CONTROLLER

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using Leap.Unity.Attributes;
using System.IO;
using System;
using UnityEngine.SceneManagement;


namespace Leap.Unity
{
    public class GameController : MonoBehaviour
```

```
{

    public GameObject shot;
    public Transform shotSpawn;
    public PlayerController playerController;
    public float fireRate;
    private float nextFire;

    public GameObject hazard;
    public Vector3 spawnValues;
    public int hazardCount;
    public float spawnWait;
    public float startWait;
    public float waveWait;

    public Text scoreText;
    public Text restartText;
    public Text gameOverText;
    public Text nameText;
    public Text startText;
    public Text resultadosText;

    private bool gameOver;
    private bool restart;

    private bool nameBool;
    private bool playBool;
    private bool loadBool;

    private float asteroides;
    private float fallos;

    private string nombre;
    private int score;

    DateTime tiempoInicio;
    DateTime tiempoFinal;
    float tiempoTotal;

    private float horda;

    float horizontalMax , horizontalMin, verticalMax ,verticalMin;


    [Units("seconds")]
    [Tooltip("The interval in seconds at which to check this detector's conditions.")]
    [MinValue(0)]
    public float Period = .1f; //seconds

    [Tooltip("The hand model to watch. Set automatically if detector is on a hand.")]
    public IHandModel HandModelRight = null;
    public IHandModel HandModelLeft = null;
    IHandModel HandModel = null;

    Hand hand;

    Vector3 palmVector;
    Vector3 palmVector2;
    Vector2 palma;
    Vector2 palma2;
    Vector2 horizonte;
    Vector2 horizonte2;
    float angle;  // Movimiento vertical
    float angle2; // Movimiento horizontal


    public Canvas empezarCanvas;
    bool empezarBool;
    public Slider hazardSlider;

    public Canvas finalCanvas;
    bool reiniciarBool;
    bool salirBool;
```

```
string Nombre;
string NombreMano;

public Slider velocidadAsteroide;

void UpdateScore()
{
    scoreText.text = "Score: " + score;
}


void Start()
{
    gameOver = false;
    restart = false;
    nameBool = true;
    loadBool = false;
    playBool = false;
    restartText.text = "";
    gameOverText.text = "";
    startText.text = "";
    nameText.text = "";
    score = 0;
    horda = 0;
    fallos = 0;

    empezarCanvas.enabled = true;
    empezarBool = false;
    finalCanvas.enabled = false;
    reiniciarBool = false;
    salirBool = false;



    UpdateScore();

    Cargar2();
    Cargar();

    if (NombreMano == "Izquierda")
    {
        HandModel = HandModelLeft;

    }
    else if (NombreMano == "Derecha")
    {
        HandModel = HandModelRight;

    }


}
void FixedUpdate()
{
    Moverse();

}

void Update()
{

    if (HandModel != null)

    {
        hand = HandModel.GetLeapHand();
        if (hand != null)
        {
            palmVector = hand.PalmNormal.ToVector3();
            palmVector2 = hand.Direction.ToVector3();
            horizonte.Set(1, 0);
            palma.Set(palmVector.z, palmVector.y);
            angle = (Vector2.Angle(horizonte, palma) - 90) * -1;
            horizonte2.Set(1, 0);
```

```
                    palma2.Set(palmVector2.x, palmVector2.z);
                    angle2 = (Vector2.Angle(palma2, horizonte2) - 90);

                }
                else
                {
                    angle = 0;
                    angle2 = 0;
                }




            }

            if (restart)
            {
                if (reiniciarBool)
                {
                    SceneManager.LoadScene("Main");
                }
                if(salirBool)
                {
                    string seguido;
                    seguido =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
                    StreamReader read = File.OpenText(seguido);
                    string leer;
                    leer = read.ReadLine().ToString();

                    if (leer == "a")
                    {
                        SceneManager.LoadScene("Topos");

                    }
                    else
                    {
                        SceneManager.LoadScene("Menu_Inicial");
                    }

                }
            }
        }

        IEnumerator SpawnWaves()
        {
                yield return new WaitForSeconds(startWait);

            while (true)
                {
                    for (int i = 0; i < asteroides; i++)
                    {
                        Vector3 spawnPosition = new Vector3(UnityEngine.Random.Range(-
spawnValues.x, spawnValues.x), spawnValues.y, spawnValues.z);
                        Quaternion spawnRotation = Quaternion.identity;
                    GameObject clone;


                    clone = Instantiate(hazard, spawnPosition, spawnRotation);
                    clone.GetComponent<Rigidbody>().velocity = -(transform.forward *
velocidadAsteroide.value * 2);
                    yield return new WaitForSeconds(spawnWait);
                    }
                    horda++;
                    yield return new WaitForSeconds(waveWait);

                    if (asteroides == 0)
                    {

                    tiempoFinal = DateTime.Now;
                    TimeSpan tiempo = tiempoFinal - tiempoInicio;
                    tiempoTotal = tiempo.Seconds;
```

```
                    resultadosText.text = "TUS RESULTADOS: \n Golpes a la nave: " + fallos +
"\n Tiempo: " + tiempoTotal + " segundos";
                    finalCanvas.enabled = true;
                    restart = true;
                    Guardar();

                    break;
                    }
                }
        }



        public void Asteroide_Fallar()
        {
            fallos++;
        }

        public void Asteroide_Destruir()
        {
            asteroides = asteroides - 1;
        }

        public void AddScore(int newScoreValue)
        {
            score += newScoreValue;
            UpdateScore();
        }

        public void GameOver()
        {
            gameOverText.text = "Game Over!";
            gameOver = true;
        }

        void Cargar2()
        {
            string guardadoNombre;
            guardadoNombre =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado_Nombre\\Nombre.doc
";
            StreamReader read = File.OpenText(guardadoNombre);
            Nombre = read.ReadLine();
            NombreMano = read.ReadLine();
            read.Close();
        }

        void Cargar()
        {
            horizontalMax = 0;
            horizontalMin = 0;
            verticalMax = 0;
            verticalMin = 0;

            char[] leer = new char[40];

                string guardado;
                guardado =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado_angulos_muneca\\"
+ Nombre + ".doc";


                if (File.Exists(guardado))
                {

                    StreamReader read = File.OpenText(guardado);
                    //this is to go only to the last 4 lines.


                    //we do things to know how to charge only the angle
                    // 3rd line is horizontalMax angle
                    leer = read.ReadLine().ToCharArray();
```

```
                    //We take only 3 decimals not 5 that have the evaluation to simplificate
                    for (int i = 0; i < 6; i++)
                    {

                        if (i < 2)
                        {
                            horizontalMax = horizontalMax + (leer[i] - 48) *
(float)Math.Pow(10, (1 - i));
                        }
                        if (i > 2)
                        {
                            horizontalMax = horizontalMax + (leer[i] - 48) *
(float)Math.Pow(10, (2 - i));
                        }
                    }



                    //4th line is horizontalMin angle
                    leer = read.ReadLine().ToCharArray();
                    for (int i = 0; i < 6; i++)
                    {

                        if (i < 2)
                        {
                            horizontalMin = horizontalMin - (leer[i+1] - 48) *
(float)Math.Pow(10, (1 - i));
                        }
                        if (i > 2)
                        {
                            horizontalMin = horizontalMin - (leer[i+1] - 48) *
(float)Math.Pow(10, (2 - i));
                        }
                    }


                    //5th line is verticalMax angle
                    leer = read.ReadLine().ToCharArray();
                    for (int i = 0; i < 6; i++)
                    {

                        if (i < 2)
                        {
                            verticalMax = verticalMax + (leer[i] - 48) * (float)Math.Pow(10,
(1 - i));
                        }
                        if (i > 2)
                        {
                            verticalMax = verticalMax + (leer[i] - 48) * (float)Math.Pow(10,
(2 - i));
                        }
                    }


                    //6th line is verticalMin angle
                    leer = read.ReadLine().ToCharArray();
                    for (int i = 0; i < 6; i++)
                    {

                        if (i < 2)
                        {
                            verticalMin = verticalMin - (leer[i+1] - 48) *
(float)Math.Pow(10, (1 - i));
                        }
                        if (i > 2)
                        {
                            verticalMin = verticalMin - (leer[i+1] - 48) *
(float)Math.Pow(10, (2 - i));
                        }
                    }

                    read.Close();
            }
```

```
        }

        public void DispararLeft()
        {
                if (Time.time > nextFire)
                {
                    nextFire = Time.time + fireRate;
                    Instantiate(shot, shotSpawn.position, shotSpawn.rotation); // as
GameObject;
                    GetComponent<AudioSource>().Play();
                }
        }

        public void DispararRight()
        {
                if (Time.time > nextFire)
                {
                    nextFire = Time.time + fireRate;
                    Instantiate(shot, shotSpawn.position, shotSpawn.rotation); // as
GameObject;
                    GetComponent<AudioSource>().Play();
                }
        }

        void Moverse()
        {
            playerController.Moverse(horizontalMax, horizontalMin, verticalMax, verticalMin,
angle, angle2);
        }



        void Guardar()
        {
         DateTime fecha = DateTime.Now;
         string guardado;

        guardado =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado_naves\\" + Nombre
+ ".doc";

            if (!File.Exists(guardado))
            {
             StreamWriter write = File.CreateText(guardado);
             write.Write(
                       "Fecha: " + fecha.ToString() + "\n"
                       + "Nombre: " + nombre + "\n"
                       + "Horda: " + horda + "\n"
                       + "Puntuación: " + score + "\n . \n . \n");

             write.Flush();
             write.Close();
            }

            else
            {
                StreamWriter write = File.AppendText(guardado);
             write.Write(
                         "Fecha: " + fecha.ToString() + "\n"
                    + "Nombre: " + nombre + "\n"
                    + "Horda: " + horda + "\n"
                    + "Puntuación: " + score + "\n . \n . \n");
                write.Flush();
                write.Close();
            }
        }

        public void EmpezarButton()
        {
            empezarBool = true;
            empezarCanvas.enabled = false;
            asteroides = hazardSlider.value;
            tiempoInicio = DateTime.Now;
            StartCoroutine(SpawnWaves());
```

```
        }

        public void SalirButton()
        {
            salirBool = true;
            finalCanvas.enabled = false;
        }

        public void ReiniciarButton()
        {
            reiniciarBool = true;
            finalCanvas.enabled = false;
        }
    }
}
```

8.4.5. MOVER

```
using UnityEngine;

public class Mover : MonoBehaviour {

    public float speed;

     void Start()
    {
        GetComponent<Rigidbody>().velocity = transform.forward * speed;
    }

}
```

8.4.6. MOVER ASTEROIDE

```
using UnityEngine;
using UnityEngine.UI;

public class Mover_Asteroide : MonoBehaviour {

    public Slider speedSlider;
    float speed;

    void Start()
    {
        speed = speedSlider.value;
        GetComponent<Rigidbody>().velocity =  - (transform.forward * speed * 2) ;
    }
}
```

8.4.7. PLAYER CONTROLLER

```
using UnityEngine;

[System.Serializable]
public class Boundary
{
    public float xMin, xMax, zMin, zMax;
}

public class PlayerController : MonoBehaviour {

    public float speed;
    public Boundary boundary;
    public float tilt;
    public float verticalRange;
    public float horizontalRange;

    public GameObject shot;
    public Transform shotSpawn;

    public float fireRate;
    private float nextFire;
```

```
    public void Moverse(float horizontalMax, float horizontalMin, float verticalMax, float
verticalMin, float angle,float angle2)
    {
        float horizontal, vertical;

        if (angle>0 && (angle >= (horizontalMax -verticalRange)))
        {
            vertical = 1;
        }

        else if(angle<0 && (angle <= (horizontalMin + verticalRange)))
        {
            vertical = -1;
        }

        else
        {
            vertical = 0;
        }

        if (angle2 > 0 && (angle2 >= (verticalMax - horizontalRange)))
        {
            horizontal = -1;
        }

        else if (angle2 < 0 && (angle2 <= (verticalMin + horizontalRange)))
        {
            horizontal = 1;
        }

        else
        {
            horizontal = 0;
        }

        Vector3 movement = new Vector3(horizontal, 0.0f, vertical);
        GetComponent<Rigidbody>().velocity = movement * speed;

        GetComponent<Rigidbody>().position = new Vector3
            (
                Mathf.Clamp(GetComponent<Rigidbody>().position.x, boundary.xMin,
boundary.xMax),
                0.0f,
                Mathf.Clamp(GetComponent<Rigidbody>().position.z, boundary.zMin,
boundary.zMax)


            );


        GetComponent<Rigidbody>().rotation = Quaternion.Euler(0.0f, 0.0f,
GetComponent<Rigidbody>().velocity.x * -tilt);

    }

}
```

8.4.8. RANDOM ROTATION

```
using UnityEngine;

public class RandomRotation : MonoBehaviour {

    public float tumble;

    void Start()
    {
        GetComponent<Rigidbody>().angularVelocity = Random.insideUnitSphere * tumble;

    }
}
```

8.5. TOPOS

8.5.1. BOTONES SLIDERS

```
using UnityEngine;
namespace Leap.Unity
{
    public class BotonesSliders : MonoBehaviour
    {

        public GameController2 gameController;


        public void empezarButton()
        {
            gameController.empezarButton();
        }

    }
}
```

8.5.2. DESTROY BY BOUNDARY 2

```
using UnityEngine;
namespace Leap.Unity
{
    public class DestroyByBoundary2 : MonoBehaviour
    {

        public GameController2 gameController;


        void OnTriggerStay(Collider other)
        {

            if (other.tag == "Topo" && (gameController.Cazar_Topo() ||
gameController.Siguiente()))
            {
                Destroy(other.gameObject);
                gameController.Guardar_Topo();

                gameController.TopoBool_false();
                gameController.ContinuarBool_False();

            }

            if (other.tag == "Manzana" && (gameController.Coger_Manzana() ||
gameController.Siguiente()))
            {

                Destroy(other.gameObject);
                gameController.Guardar_Manzana();
                gameController.ManzanaBool_false();
                gameController.ContinuarBool_False();
            }

        }

    }
}
```

8.5.3. DESTROY BY CONTACT

```
using UnityEngine;

    public class DestroyByContact : MonoBehaviour
    {

        void OnTriggerEnter(Collider other)
```

```
        {
            if (other.tag == "BolaTopo")
            {
                Destroy(gameObject);
                Destroy(other.gameObject);
            }
            if (other.tag == "BolaManzana")
            {
                Destroy(gameObject);
                Destroy(other.gameObject);
            }

        }
    }
```

8.5.4. GAME CONTROLLER 2

```
using System;
using UnityEngine;
using UnityEngine.UI;
using Leap.Unity.Attributes;
using System.IO;
using UnityEngine.SceneManagement;
namespace Leap.Unity
{
    public class GameController2 : MonoBehaviour
    {


        [Units("seconds")]
        [Tooltip("The interval in seconds at which to check this detector's conditions.")]
        [MinValue(0)]
        public float Period = .1f; //seconds

        [Tooltip("The hand model to watch. Set automatically if detector is on a hand.")]
        public IHandModel HandModelRight = null;
        public IHandModel HandModelLeft = null;
        IHandModel HandModel = null;

        Hand hand;

        // the bools are as states
        private bool inicio;
        private bool proximalBool;
        private bool medioBool;
        private bool final;

        Vector2 bone;
        Vector2 horizonte;

        Finger[] prueba = new Finger[5];
        Vector3[] a = new Vector3[5];

        Bone[] proximal = new Bone[5];
        Bone[] medio = new Bone[5];
        Vector3[] proximalVector = new Vector3[5];
        Vector3[] medioVector = new Vector3[5];
        float[] proximalAngle = new float[5];
        float[] medioAngle = new float[5];

        float[] maxProximalAngle = new float[5];
        float[] saveProximalAngle = new float[5];
        float[] maxMedioAngle = new float[5];
        float[] saveMedioAngle = new float[5];



        Vector3[] a2 = new Vector3[5];
        string[] a2String = new string[5];

        DateTime fecha;

        float color;
```

```
public GameObject manzana;
public GameObject topo;

bool topoBool, manzanaBool;

float numManzana, numTopo;
DateTime inicioManzana, inicioTopo, finalManzana, finalTopo;
float[] tiempoManzana = new float[10];
float[] tiempoTopo = new float[10];



public float cantidadManzanas, cantidadTopos;

public Canvas empezarCanvas;
public Slider manzanasSlider;
public Slider toposSlider;
bool empezarBool;
bool finalBool;

public Text finalText;
public Canvas finalCanvas;
public Canvas juegoCanvas;
bool continuarBool;

string Nombre;
string NombreMano;

public Text pruebaText;

int fallos;


void Start()
{
    inicio = true;
    proximalBool = false;
    medioBool = false;
    final = false;
    finalBool = false;
    topoBool = false;
    manzanaBool = false;
    continuarBool = false;
    for (int i = 1; i < 5; i++)
    {
        maxMedioAngle[i] = 0;
        saveMedioAngle[i] = 0;
        maxProximalAngle[i] = 0;
        saveProximalAngle[i] = 0;
        medioAngle[i] = 35;
        proximalAngle[i] = 35;
    }

    numManzana = 0;
    numTopo = 0;

    fallos = 0;

    for (int i = 0; i <10; i++)
    {
        tiempoManzana[i] = 0;
        tiempoTopo[i] = 0;


    }

    empezarCanvas.enabled = true;
    finalText.text = "";
    finalCanvas.enabled = false;
    juegoCanvas.enabled = false;
    Cargar2();
    Cargar();

    if (NombreMano == "Izquierda")
```

```
            {
                HandModel = HandModelLeft;
            }
            else if (NombreMano == "Derecha")
            {
                HandModel = HandModelRight;
            }
        }

        void Update()
        {
            if (HandModel != null)
            {
                hand = HandModel.GetLeapHand();
                if (hand != null)
                {
                    for (int i = 0; i < 5; i++)
                    {
                        prueba[i] = hand.Fingers[i];
                        horizonte.Set(1, 0);
                        if (i > 0)
                        {
                            proximal[i] = prueba[i].bones[1];
                            proximalVector[i] = proximal[i].Direction.ToVector3();
                            bone.Set(proximalVector[i].z, proximalVector[i].y);
                            proximalAngle[i] = (Vector2.Angle(horizonte, bone));
                            medio[i] = prueba[i].bones[2];
                            medioVector[i] = medio[i].Direction.ToVector3();
                            bone.Set(medioVector[i].z, medioVector[i].y);
                            medioAngle[i] = (Vector2.Angle(horizonte, bone));
                        }
                    }
                }

            }

            if (empezarBool)
            {
                Salir_Esfera();

            }
            if(finalBool)
            {
                string seguido;
                seguido =
"C:/Users/Usuario/Desktop/TFG/ejemplos_de_tutoriales/Menu/Guardado_Seguido/Seguido.doc";
                if (File.Exists(seguido))
                {
                    StreamWriter write2 = File.CreateText(seguido);
                    write2.Write("b");
                    write2.Flush();
                    write2.Close();
                }
                Guardar();
                SceneManager.LoadScene("Menu_Inicial");
            }

            if(Coger_Manzana())
            {
                pruebaText.text = "Posición Coger Manzana";
            }
            else if (Cazar_Topo())
            {
                pruebaText.text = "Posición Cazar Cerdo";
            }
            else if (Mano_Extendida())
            {
                pruebaText.text = "Posición Mano Abierta";
            }
            else
            {
                pruebaText.text = "";
            }
```

```
}
void Salir_Esfera()
{
    if (topoBool == false && manzanaBool == false && Mano_Extendida())
    {
        color = UnityEngine.Random.value;
        if (numTopo >= toposSlider.value && numManzana >= manzanasSlider.value)
        {
            juegoCanvas.enabled = false;
            finalCanvas.enabled = true;

            finalText.text = "El numero de manzanas ha sido: " + numManzana +
                "\n El numero de cerdos ha sido: " + numTopo +
                "\n El numero de fallos ha sido: " + fallos;

        }
        else if (numTopo >= toposSlider.value)
        {
            Salir_Manzana();
        }
        else
        {
            if (color > 0.5 || numManzana >= manzanasSlider.value)
            {
                Salir_Topo();
            }
            else
            {
                Salir_Manzana();
            }
        }
    }
}



void Salir_Topo()
{

    Instantiate(topo);
    topoBool = true;
    numTopo++;
    inicioTopo = DateTime.Now;

}

void Salir_Manzana()
{
    Instantiate(manzana);
    manzanaBool = true;
    numManzana++;
    inicioManzana = DateTime.Now;

}


public void ManzanaBool_false()
{
    manzanaBool = false;

}

public void TopoBool_false()
{
    topoBool = false;


}

public void Guardar_Manzana()
{
    finalManzana = DateTime.Now;
```

```
            TimeSpan duracion = finalManzana - inicioManzana;
            bool manzanaFlag = true;
            for (int i = 0; i < 10; i++)
            {
                if (tiempoManzana[i] == 0 && manzanaFlag && continuarBool == false)
                {
                    tiempoManzana[i] = duracion.Milliseconds;
                    manzanaFlag = false;

                }
            }


        }

        public void Guardar_Topo()
        {
            finalTopo = DateTime.Now;
            TimeSpan duracion = finalTopo - inicioTopo;
            bool topoFlag = true;

            for(int i = 0; i <10; i++)
            {
                if (tiempoTopo[i] == 0 && topoFlag && continuarBool == false)
                {
                    tiempoTopo[i] = duracion.Milliseconds;
                    topoFlag = false;

                }
            }
        }

        void Guardar()
        {
            string guardado;

            guardado =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado\\" + Nombre +
".csv";
            StreamWriter write = File.AppendText(guardado);
            bool guardarFallos = true;

            for(int i = 0; i < 10; i++)
            {
                if(tiempoTopo[i] != 0 && tiempoManzana[i] != 0 )
                {
                    write.Write(", , , , , , , , , , , , , , , , , , , ," +
tiempoManzana[i] + ", ," + tiempoTopo[i] + "\n");
                }
                else if( tiempoTopo[i] != 0)
                {
                    write.Write(", , , , , , , , , , , , , , , , , , , ," +
tiempoTopo[i] + "\n");
                }
                else if ( tiempoManzana[i] != 0)
                {
                    write.Write(", , , , , , , , , , , , , , , , , , , ," +
tiempoManzana[i] + "\n");
                }
                else if (guardarFallos)
                {
                    write.Write(", , , , , , , , , , , , , , , , , , , , , ," + fallos
+ "\n");
                    guardarFallos = false;
                }
            }

            write.Flush();
            write.Close();
        }


        public bool Cazar_Topo()
        {
```

```
            float dedosBajados = 0;
            if ((saveProximalAngle[1] -10) < proximalAngle[1] && proximalAngle[1] <
(saveProximalAngle[1] + 10) )
            {
                dedosBajados++;
            }
            if ((saveProximalAngle[2] - 10) < proximalAngle[2] && proximalAngle[2] <
(saveProximalAngle[2] + 10))
            {
                dedosBajados++;
            }
            if ((saveProximalAngle[3] - 10) < proximalAngle[3] && proximalAngle[3] <
(saveProximalAngle[3] + 10))
            {
                dedosBajados++;
            }
            if ((saveProximalAngle[4] - 10) < proximalAngle[4] && proximalAngle[4] <
(saveProximalAngle[4] + 10))
            {
                dedosBajados++;
            }
            if (dedosBajados >= 4)
            {
                return true;
            }
            else return false;

        }

        public bool Coger_Manzana()
        {


            float dedosBajados = 0;

            if ((saveMedioAngle[1] - 10) < medioAngle[1] && medioAngle[1] <
(saveMedioAngle[1] + 10))
            {
                dedosBajados++;
            }
            if ((saveMedioAngle[2] - 10) < medioAngle[2] && medioAngle[2] <
(saveMedioAngle[2] + 10))
            {
                dedosBajados++;
            }
            if ((saveMedioAngle[3] - 10) < medioAngle[3] && medioAngle[3] <
(saveMedioAngle[3] + 10))
            {
                dedosBajados++;
            }
            if ((saveMedioAngle[4] - 10) < medioAngle[4] && medioAngle[4] <
(saveMedioAngle[4] + 10))
            {
                dedosBajados++;
            }

            if (dedosBajados >= 4)
            {
                return true;
            }
            else return false;
        }

        bool Mano_Extendida()
        {
            float dedosBajados = 0;
            float huesosBajados = 0;
            if (proximalAngle[1] < 30)
            {
                dedosBajados++;
            }
            if (proximalAngle[2] < 30)
            {
                dedosBajados++;
```

```
            }
            if (proximalAngle[3] < 30)
            {
                dedosBajados++;
            }
            if (proximalAngle[4] < 30)
            {
                dedosBajados++;
            }
            if (medioAngle[1] < 30)
            {
                huesosBajados++;
            }
            if (medioAngle[2] < 30)
            {
                huesosBajados++;
            }
            if (medioAngle[3] < 30)
            {
                huesosBajados++;
            }
            if (medioAngle[4] < 30)
            {
                huesosBajados++;
            }
            if (dedosBajados >= 4 && huesosBajados >= 4)
            {
                return true;
            }
            else return false;
        }



        void Cargar2()
        {
            string guardadoNombre;
            guardadoNombre =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado_Nombre\\Nombre.doc
";
            StreamReader read = File.OpenText(guardadoNombre);
            Nombre = read.ReadLine();
            NombreMano = read.ReadLine();
            read.Close();
        }
        void Cargar()
        {
            string guardadoNombre;
            guardadoNombre =
"C:\\Users\\Usuario\\Desktop\\TFG\\ejemplos_de_tutoriales\\Menu\\Guardado_angulos_dedos\\" +
Nombre + ".doc";



                char[] leer = new char[40];
                char[] leer_angle = new char[10];



                    if (File.Exists(guardadoNombre))
                    {



                        StreamReader read = File.OpenText(guardadoNombre);

                for (int j = 1; j < 5; j++)
                {
                    leer = read.ReadLine().ToCharArray();
                    float hasta;

                    if(leer[3] == '.')
                    {
```

```
                        hasta = 3;

                    }
                    else
                    {
                        hasta = 2;
                    }

                    for (int i = 0; i < 6; i++)
                    {

                        if (i < hasta)
                        {
                            saveProximalAngle[j] = saveProximalAngle[j] + (leer[i] - 48) *
(float)Math.Pow(10, ((hasta-1) - i));
                        }
                        if (i > hasta)
                        {
                            saveProximalAngle[j] = saveProximalAngle[j] + (leer[i] - 48) *
(float)Math.Pow(10, (hasta - i));
                        }
                    }

                }
                for (int j = 1; j < 5; j++)
                {

                    leer = read.ReadLine().ToCharArray();
                    float hasta;

                    if (leer[3]  == '.')
                    {
                        hasta = 3;
                    }
                    else
                    {
                        hasta = 2;
                    }

                    for (int i = 0; i < 6; i++)
                    {

                        if (i < 2)
                        {
                            saveMedioAngle[j] = saveMedioAngle[j] + (leer[i] - 48) *
(float)Math.Pow(10, ((hasta -1) - i));
                        }
                        if (i > 2)
                        {
                            saveMedioAngle[j] = saveMedioAngle[j] + (leer[i] - 48) *
(float)Math.Pow(10, (hasta - i));
                        }
                    }
                }
                        read.Close();
                    }

            }


        public void empezarButton()
        {
            if (manzanasSlider.value != 0 || toposSlider.value != 0)
            {
                empezarBool = true;
                empezarCanvas.enabled = false;
                juegoCanvas.enabled = true;
            }
        }

        public void FinalizarButton()
        {
            finalBool = true;
        }
```

```
        public void ContinuarButton()
        {

            continuarBool = true;
            fallos++;
        }

        public bool Siguiente()
        {
            return continuarBool;
        }
        public void ContinuarBool_False()
        {
            continuarBool = false;
        }
    }

}
```